Cogent DataHub®
Version 9.0

A memory resident real-time database that acts as a hub, providing fast and efficient concentration and distribution of data for OPC and other Windows applications.

**Quick Links:**

OPC UA | OPC DA | OPC A&E | Tunnelling | Bridging | Redundancy | Write to a Database | Query a Database | Web Server | WebView | MQTT Client | MQTT Broker | Modbus | Email/SMS | Camera | Remote Config | Scripting | Security

Data Browser | Connection Viewer | Event Log | Script Log

Copyright © Skkynet. All rights reserved. Legal notices
# Table of Contents

## Installation
- System Requirements and Installation ................................................................. 1
- Installing Licenses ................................................................................................... 1
- Installing the DataHub as a Service ................................................................. 2
- Performing a Silent (Unattended) Install .............................................................. 3
- Upgrading to a new version .................................................................................. 3
- Installing Version 9.x over previous versions ....................................................... 4
- Downgrading Version 9.x to a previous version ..................................................... 4

## Configuration Files
- Backing Up or Moving a DataHub Installation .................................................. 8

## Getting Started
- Running the Cogent DataHub ............................................................................. 10
- Test with simulated data ..................................................................................... 13
- Connect to an OPC DA server ........................................................................... 14
- Connect from an OPC DA client ......................................................................... 22
- Test MQTT .......................................................................................................... 23
- Connect to remote data ...................................................................................... 25
- Custom Connections ............................................................................................ 27
- Performance Limitations ...................................................................................... 27
- Reporting Problems ............................................................................................. 30

## Properties Window
- For All Options ..................................................................................................... 34
- General ................................................................................................................... 35
- OPC UA ................................................................................................................ 36
- OPC DA ................................................................................................................ 53
- OPC A&E .............................................................................................................. 62
- Tunnel/Mirror ....................................................................................................... 66
- Bridging .................................................................................................................. 72
- Redundancy .......................................................................................................... 76
- Database ............................................................................................................... 81
- Web Server ......................................................................................................... 86
- DataHub WebView ............................................................................................... 88
- MQTT Client ........................................................................................................ 90
- Make the Connection ......................................................................................... 90
- Exchange Data ..................................................................................................... 96
- Pre-Configured Connections ............................................................................ 99
- MQTT Broker ...................................................................................................... 103
- Modbus ............................................................................................................... 107
- Historian .............................................................................................................. 118
- Email/SMS ............................................................................................................ 122
- Camera ............................................................................................................... 124
- System Monitor .................................................................................................. 126
- DDE ...................................................................................................................... 129
Streaming XML How-To ................................................................. 302
Polling XML How-To ................................................................. 308
ASP File Locations ................................................................. 310
Using MQTT ................................................................................... 313
Connecting MQTT Clients with the DataHub MQTT Broker ............... 313
Making MQTT Client Connections ................................................. 317
Security ....................................................................................... 326
How to Configure ........................................................................ 326
SSL and Firewalls ....................................................................... 329
User Authentication ..................................................................... 331
Authorization and User Permissions .............................................. 333
Permissions for the DataHub Command Set ................................... 336
Passwords .................................................................................. 339
Email and SMS ........................................................................... 341
Introduction ................................................................................ 341
Configuring the Mail Server ......................................................... 342
Sending a Test Message ............................................................... 343
Defining the Email Message ......................................................... 347
Assigning a Trigger ..................................................................... 349
Setting Trigger Conditions .......................................................... 351
Configured Actions ..................................................................... 355
Sending SMS Text Messages ......................................................... 356
HTML Message Examples ............................................................ 357
An HTML Message with Embedded Data Points ................................. 357
An HTML Message with a Table Created in Code ............................. 359
Dynamically Changing Email Subjects and Recipients ....................... 361
System Monitor ........................................................................... 363
Introduction ................................................................................... 363
Configuring the System Monitor ..................................................... 363
Monitoring Systems Across a Network ............................................ 366
Excel Connections ....................................................................... 372
Getting Data into Excel ............................................................... 372
Method 1 - Drag and Drop using DDEAdvise ................................. 372
Method 2 - Excel Macros using DDERequest ................................... 375
Getting Data out of Excel ............................................................ 377
Method 1 - Configuring DDEAdvise loops in the Cogent DataHub .......... 377
Method 2 - Writing Excel macros that use the DDDEPoke command ......... 380
Networking Excel ........................................................................ 382
Working with Ranges ................................................................... 388
Getting a Range out of Excel ....................................................... 388
Getting a Range into Excel .......................................................... 389
Sample Excel Macros for Arrays .................................................... 392
Basic Trouble-Shooting for Excel Connections ................................. 394
DataHub Scripting ....................................................................... 398
Tools ........................................................................................ 398
DataHub ODBC (Open Database Connectivity) Scripting ........................... 399
DataHub Windows Scripting ................................................................. 400
Working With Data .................................................................................. 401
Data Points ............................................................................................. 401
  Creating New Points .............................................................................. 401
  Deleting Points .................................................................................... 401
  Viewing Data Points ............................................................................. 402
  Point Size Limits ................................................................................ 402
  Data Types .......................................................................................... 402
Data Communication Concepts ............................................................... 403
  Send and Receive Data ......................................................................... 403
  Client - Server ..................................................................................... 404
  Synchronous and Asynchronous Communication .................................. 404
Data Exchange Protocols ......................................................................... 404
  OPC Protocol ....................................................................................... 404
  DDE Protocol ....................................................................................... 406
  Tunnelling/Mirroring .......................................................................... 406
  DHTP - The DataHub Transfer Protocol ............................................. 407
  The DataHub API ................................................................................ 408
Data Organization ..................................................................................... 408
  Data Domains ....................................................................................... 409
  Assemblies, Subassemblies, Attributes, and Properties ....................... 409
  Attributes and Types ........................................................................... 410
  Example 1: Attributes and Types ....................................................... 411
  Example 2: Private Attributes ............................................................ 413
Optimizing Data Throughput ................................................................. 415
  Binary Mode Tunnel/Mirror (TCP) Connections ................................... 415
  Tunnel/Mirror (TCP) Heartbeat and Timeout ....................................... 415
  Old Value Queuing .............................................................................. 416
  Un-Buffered Delivery ......................................................................... 417
  Screen Output ....................................................................................... 417
  CPU Saturation ..................................................................................... 417
  How to Optimize ................................................................................ 418
  Tunnel/Mirror (TCP) connections ....................................................... 418
  DataHub C++ API .............................................................................. 420
  Gamma scripts ................................................................................... 420
Using DataHub Commands ...................................................................... 421
  Command Syntax ............................................................................... 421
  Return Syntax ..................................................................................... 421
  Sending Commands by TCP ............................................................. 422
I. Cogent DataHub Command Set .......................................................... 424
  acksuccess ......................................................................................... 425
  add ................................................................................................. 426
  alias ............................................................................................... 427
  alive ............................................................................................... 428
  append ............................................................................................ 429
  assembly ......................................................................................... 430
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>431</td>
</tr>
<tr>
<td>auth</td>
<td>432</td>
</tr>
<tr>
<td>authreload</td>
<td>433</td>
</tr>
<tr>
<td>auto_create_domains</td>
<td>434</td>
</tr>
<tr>
<td>auto_timestamp</td>
<td>435</td>
</tr>
<tr>
<td>bridge</td>
<td>436</td>
</tr>
<tr>
<td>bridge_remove</td>
<td>438</td>
</tr>
<tr>
<td>bridge_remove_pattern</td>
<td>439</td>
</tr>
<tr>
<td>bridge_transform</td>
<td>440</td>
</tr>
<tr>
<td>cforce</td>
<td>442</td>
</tr>
<tr>
<td>cread</td>
<td>444</td>
</tr>
<tr>
<td>create</td>
<td>446</td>
</tr>
<tr>
<td>create_domain</td>
<td>447</td>
</tr>
<tr>
<td>creport</td>
<td>448</td>
</tr>
<tr>
<td>cset</td>
<td>449</td>
</tr>
<tr>
<td>cwrite</td>
<td>451</td>
</tr>
<tr>
<td>DDEAdvise</td>
<td>453</td>
</tr>
<tr>
<td>DDEConnect</td>
<td>454</td>
</tr>
<tr>
<td>DDEDisconnect</td>
<td>455</td>
</tr>
<tr>
<td>DDEExcelUnicode</td>
<td>456</td>
</tr>
<tr>
<td>DDEInit</td>
<td>457</td>
</tr>
<tr>
<td>DDEServe</td>
<td>458</td>
</tr>
<tr>
<td>DDEUnadvise</td>
<td>459</td>
</tr>
<tr>
<td>DDEUnadvisePattern</td>
<td>460</td>
</tr>
<tr>
<td>DDEUnadvisePoint</td>
<td>461</td>
</tr>
<tr>
<td>debug</td>
<td>462</td>
</tr>
<tr>
<td>defaultprop</td>
<td>463</td>
</tr>
<tr>
<td>delete</td>
<td>464</td>
</tr>
<tr>
<td>div</td>
<td>465</td>
</tr>
<tr>
<td>domain</td>
<td>466</td>
</tr>
<tr>
<td>domain_bridge</td>
<td>467</td>
</tr>
<tr>
<td>domain_bridge_enable</td>
<td>469</td>
</tr>
<tr>
<td>domain_bridge_prefer</td>
<td>470</td>
</tr>
<tr>
<td>domain_bridge_refresh</td>
<td>471</td>
</tr>
<tr>
<td>domain_bridge_remove</td>
<td>472</td>
</tr>
<tr>
<td>domains</td>
<td>473</td>
</tr>
<tr>
<td>dump</td>
<td>474</td>
</tr>
<tr>
<td>enable_bridging</td>
<td>475</td>
</tr>
<tr>
<td>enable_dde_client</td>
<td>476</td>
</tr>
<tr>
<td>enable_dde_server</td>
<td>477</td>
</tr>
<tr>
<td>enable_domain_bridging</td>
<td>478</td>
</tr>
<tr>
<td>enable_opc_client</td>
<td>479</td>
</tr>
<tr>
<td>enable_opc_server</td>
<td>480</td>
</tr>
<tr>
<td>enable_scripting</td>
<td>481</td>
</tr>
<tr>
<td>error</td>
<td>482</td>
</tr>
<tr>
<td>execute_plugin</td>
<td>483</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>exit</td>
<td>484</td>
</tr>
<tr>
<td>force</td>
<td>485</td>
</tr>
<tr>
<td>format</td>
<td>487</td>
</tr>
<tr>
<td>get_client_stats</td>
<td>488</td>
</tr>
<tr>
<td>heartbeat</td>
<td>490</td>
</tr>
<tr>
<td>HistorianAdd</td>
<td>491</td>
</tr>
<tr>
<td>HistorianFlags</td>
<td>492</td>
</tr>
<tr>
<td>HistorianRemove</td>
<td>493</td>
</tr>
<tr>
<td>HistorianSaveConfig</td>
<td>494</td>
</tr>
<tr>
<td>HistorianSetConfiguring</td>
<td>495</td>
</tr>
<tr>
<td>HistoryGroupAdd</td>
<td>496</td>
</tr>
<tr>
<td>HistoryGroupAddPoint</td>
<td>497</td>
</tr>
<tr>
<td>HistoryGroupDeadband</td>
<td>498</td>
</tr>
<tr>
<td>HistoryGroupDefault</td>
<td>499</td>
</tr>
<tr>
<td>HistoryGroupFileTimes</td>
<td>500</td>
</tr>
<tr>
<td>HistoryGroupFlushTimes</td>
<td>501</td>
</tr>
<tr>
<td>HistoryGroupRemove</td>
<td>502</td>
</tr>
<tr>
<td>HistoryGroupStorageTimes</td>
<td>503</td>
</tr>
<tr>
<td>ignore</td>
<td>504</td>
</tr>
<tr>
<td>ignore_old_data</td>
<td>505</td>
</tr>
<tr>
<td>include</td>
<td>506</td>
</tr>
<tr>
<td>instance</td>
<td>507</td>
</tr>
<tr>
<td>load_config_files</td>
<td>508</td>
</tr>
<tr>
<td>load_plugin</td>
<td>509</td>
</tr>
<tr>
<td>load_scripts</td>
<td>510</td>
</tr>
<tr>
<td>lock</td>
<td>511</td>
</tr>
<tr>
<td>log_file</td>
<td>512</td>
</tr>
<tr>
<td>log_file_max</td>
<td>513</td>
</tr>
<tr>
<td>log_to_file</td>
<td>514</td>
</tr>
<tr>
<td>mirror_master</td>
<td>515</td>
</tr>
<tr>
<td>mirror_master_2</td>
<td>516</td>
</tr>
<tr>
<td>ModbusApplySettings</td>
<td>519</td>
</tr>
<tr>
<td>ModbusCancelSettings</td>
<td>520</td>
</tr>
<tr>
<td>ModbusCreateSlave</td>
<td>521</td>
</tr>
<tr>
<td>ModbusDeleteSlave</td>
<td>522</td>
</tr>
<tr>
<td>ModbusEnableMaster</td>
<td>523</td>
</tr>
<tr>
<td>ModbusEnableSlave</td>
<td>524</td>
</tr>
<tr>
<td>ModbusQuerySlave</td>
<td>525</td>
</tr>
<tr>
<td>ModbusReloadSettings</td>
<td>526</td>
</tr>
<tr>
<td>ModbusSlaveAddPoint</td>
<td>527</td>
</tr>
<tr>
<td>ModbusSlaveAddRange</td>
<td>530</td>
</tr>
<tr>
<td>ModbusSlaveDeletePoint</td>
<td>533</td>
</tr>
<tr>
<td>mult</td>
<td>534</td>
</tr>
<tr>
<td>OPCActivate</td>
<td>535</td>
</tr>
<tr>
<td>OPCAddItem2</td>
<td>536</td>
</tr>
<tr>
<td>OPCAEAttach</td>
<td>538</td>
</tr>
</tbody>
</table>
OPCAEDetach ........................................................................................................ 540
OPCAEnable ........................................................................................................... 541
OPCAEnableClient ............................................................................................... 542
OPCAEnableServer ............................................................................................... 543
OPCAEFilter .......................................................................................................... 544
OPCAEServerInit ................................................................................................. 546
OPCApply ............................................................................................................... 547
OPCAAttach2 ......................................................................................................... 548
OPCCConnect ........................................................................................................ 551
OPCDetach ............................................................................................................. 552
OPCEnable ............................................................................................................. 553
OPCEnableClient ................................................................................................. 554
OPCEnableServer ................................................................................................. 555
OPCMinimumSecurity ........................................................................................... 556
OPCModify ............................................................................................................ 557
OPCQueryConnection ........................................................................................... 559
OPCQueryConnections ......................................................................................... 560
OPCQueryPoint ...................................................................................................... 561
OPCQueryPointPattern ......................................................................................... 562
OPCQueryPoints .................................................................................................... 563
OPCRefresh ........................................................................................................... 564
OPCReload ............................................................................................................. 565
OPCRemoveltem ................................................................................................. 566
private_attribute ................................................................................................. 567
property ................................................................................................................ 568
quality ................................................................................................................... 569
read ........................................................................................................................ 570
report .................................................................................................................... 572
report_domain ....................................................................................................... 573
report_errors ....................................................................................................... 574
request_initial_data ............................................................................................. 575
save_config ........................................................................................................... 576
secure .................................................................................................................... 577
set .......................................................................................................................... 578
set_access ............................................................................................................. 580
set_authoritative ................................................................................................. 581
set_canonical ........................................................................................................ 582
show_data ............................................................................................................ 583
show_debug_messages .......................................................................................... 584
show_event_log ..................................................................................................... 585
show_icon .............................................................................................................. 586
show_properties .................................................................................................. 587
show_script_log ................................................................................................. 588
subassembly ......................................................................................................... 589
success .................................................................................................................. 590
tcp_service .......................................................................................................... 591
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeout</td>
<td>592</td>
</tr>
<tr>
<td>transmit_insignificant</td>
<td>593</td>
</tr>
<tr>
<td>TunnelDelete</td>
<td>594</td>
</tr>
<tr>
<td>TunnelEnable</td>
<td>595</td>
</tr>
<tr>
<td>TunnelEnablePlain</td>
<td>596</td>
</tr>
<tr>
<td>TunnelEnableSlave</td>
<td>597</td>
</tr>
<tr>
<td>TunnelEnableSSL</td>
<td>598</td>
</tr>
<tr>
<td>TunnelPlainPort</td>
<td>599</td>
</tr>
<tr>
<td>TunnelSaveConfig</td>
<td>600</td>
</tr>
<tr>
<td>TunnelSlaveStatus</td>
<td>601</td>
</tr>
<tr>
<td>TunnelSSLCert</td>
<td>602</td>
</tr>
<tr>
<td>TunnelSSLPot</td>
<td>603</td>
</tr>
<tr>
<td>type</td>
<td>604</td>
</tr>
<tr>
<td>unload_plugin</td>
<td>605</td>
</tr>
<tr>
<td>unreport</td>
<td>606</td>
</tr>
<tr>
<td>unreport_domain</td>
<td>607</td>
</tr>
<tr>
<td>version</td>
<td>608</td>
</tr>
<tr>
<td>write</td>
<td>609</td>
</tr>
</tbody>
</table>

II. Obsolete and Unused Commands .............................................................. 611

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>asyncsocket</td>
<td>612</td>
</tr>
<tr>
<td>authgroup</td>
<td>613</td>
</tr>
<tr>
<td>authuser</td>
<td>614</td>
</tr>
<tr>
<td>bandwidth_reduce</td>
<td>615</td>
</tr>
<tr>
<td>deleted</td>
<td>616</td>
</tr>
<tr>
<td>drop_license</td>
<td>617</td>
</tr>
<tr>
<td>echo</td>
<td>618</td>
</tr>
<tr>
<td>enable_connect_server</td>
<td>619</td>
</tr>
<tr>
<td>EnableDDEServer</td>
<td>620</td>
</tr>
<tr>
<td>exception_buffer</td>
<td>621</td>
</tr>
<tr>
<td>failed_license</td>
<td>622</td>
</tr>
<tr>
<td>flush</td>
<td>623</td>
</tr>
<tr>
<td>flush_log</td>
<td>624</td>
</tr>
<tr>
<td>master_host</td>
<td>625</td>
</tr>
<tr>
<td>master_service</td>
<td>626</td>
</tr>
<tr>
<td>on_change</td>
<td>627</td>
</tr>
<tr>
<td>OPCAddItem</td>
<td>628</td>
</tr>
<tr>
<td>OPCAttach</td>
<td>630</td>
</tr>
<tr>
<td>OPCInit</td>
<td>631</td>
</tr>
<tr>
<td>point</td>
<td>632</td>
</tr>
<tr>
<td>qnx_name_attach</td>
<td>633</td>
</tr>
<tr>
<td>qnx_receiver</td>
<td>634</td>
</tr>
<tr>
<td>readid</td>
<td>635</td>
</tr>
<tr>
<td>register_datahub</td>
<td>636</td>
</tr>
<tr>
<td>report_all</td>
<td>637</td>
</tr>
<tr>
<td>report_datahubs</td>
<td>638</td>
</tr>
<tr>
<td>request</td>
<td>639</td>
</tr>
</tbody>
</table>
Appendices .................................................................................................................. 649
A. Command Line Options ................................................................................ 650
B. Excel Macro Library ....................................................................................... 653
   Configure Excel to receive data from the Cogent DataHub (using
   DDEAdvise) ........................................................................................................... 653
   Write data from Excel - User initiated (using DDEPoke) ............................ 654
   Write data from Excel - Automatically on value change (using DDEPoke) .. 655
   Other Useful Macros ....................................................................................... 657
C. Running as a Windows Service (Specified User) ............................................. 658
D. Windows Services File for Tunnel/Mirror ....................................................... 660
E. DataHub Registry Entries ............................................................................... 661
F. OPC Overview ................................................................................................ 666
G. DDE Overview ............................................................................................... 668
H. ODBC Database Concepts ............................................................................. 671
I. Error Messages .............................................................................................. 674
   Windows Error Numbers .............................................................................. 674
   Windows TCP Error Numbers ..................................................................... 675
   Windows DDE Error Numbers ................................................................... 677
J. Third-Party Source Licenses ............................................................................ 679
K. Media Source Licenses .................................................................................. 681
L. GNU General Public License .......................................................................... 682
M. GNU Lesser General Public License ............................................................. 689
Installation

System Requirements and Installation

System Requirements

The Cogent DataHub is compatible with the following versions of Windows:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Minimum Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server 2008 &amp; R2 (64-bit)</td>
<td>Service Pack 1 (SP1) + .NET 4.6.1</td>
</tr>
<tr>
<td>Windows 7 (64-bit)</td>
<td>Service Pack 1 (SP1) + .NET 4.6.1</td>
</tr>
<tr>
<td>Windows Server 2012 R2 (64-bit)</td>
<td>.NET 4.6.1</td>
</tr>
<tr>
<td>Windows 8.1 (64-bit)</td>
<td>.NET 4.6.1</td>
</tr>
<tr>
<td>Windows 10 (64-bit)</td>
<td>.NET 4.6.1</td>
</tr>
<tr>
<td>Windows Server 2016 (64-bit)</td>
<td>.NET 4.6.1</td>
</tr>
</tbody>
</table>

* If .NET 2 and .NET 4.0 are both installed, then .NET 4.5 is not required.

Installation

To install the DataHub from an archive downloaded from the Cogent web site, follow these steps:

2. Follow the instructions.

Uninstall

1. From the Start menu, select the Control Panel and then choose Add or Remove Programs.
2. Find Cogent DataHub on this list and double-click it.
3. Click the Remove button and follow the instructions.

Installing Licenses

You can install DataHub licenses purchased from Cogent using the Licenses option in the Properties window.

The Licenses option lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found on a DataHub running on any connected machine, the Cogent DataHub switches to license mode, and each connected DataHub then requires a license.
Licenses can be entered individually or loaded from a file.

- The **Enter a License...** button opens the **Enter License String** window:

Here you can paste or manually enter the text string for the license provided by Cogent. Make sure to include all colon (:) characters in the string.

The license string may contain the characters 1 (lower case L) and 1 (one) which can look nearly identical in some type fonts. If possible, it is best to copy and paste the string, rather than retyping it.

- The **Load License File...** button opens a Windows file selection window. Browse to find the directory and license file that you want to load. License files end with a .lic extension. Once you have found the license file, click the **Open** button to load the file. (Please refer to Configuration and License File Locations in the section called “Configuration Files” for more information on license file locations.)

To remove a license from your system, select one or more licenses in the **Details** window, then click the **Remove Selected** button.

**Installing the DataHub as a Service**

To install the Cogent DataHub as a service, follow these steps.

1. Start up the DataHub. (Select the program using the Windows **Start** menu, or double click the desktop icon.)
2. Configure the DataHub’s Security and Remote Config options as described in Preparation for Remote Config section of this book. You will need Remote Config to configure the DataHub while it is running as a service.
3. Test Remote Config by following the steps for Configuring a Local DataHub.
4. When you are sure that Remote Config is working properly, exit the the DataHub. (Right-click the Cogent DataHub icon in the system tray, and select **Exit** from the pop-up menu.)
5. Use the **Service Manager** to install the DataHub as a service. You should be able to
configure it using Remote Config.

If you are upgrading to a newer major version of the DataHub, please refer to Running DataHub as a service in Windows 10 and Server 2016 in Known Issues on the Cogent DataHub website for more information.

Performing a Silent (Unattended) Install

To perform an unattended install of the Cogent DataHub, you can run this command:

```
CogentDataHub-version number-date-Windows.exe /S
```

For a non-default installation directory, use:

```
CogentDataHub-version number-date-Windows.exe /S /D=C:\install\path
```

- /S indicates a silent install
- /D indicates the base installation directory

There are certain restrictions with the /D argument:

- /D must be the last argument on the line
- The path name for /D must NOT contain quotes, even if the path contains spaces.
- There must not be spaces around the = sign in the /D argument

The installation will create two directories beneath the directory indicated by /D:

- Cogent DataHub contains the Cogent DataHub installation.
- DataSim contains the two data simulators.

Upgrading to a new version

Minor Version Upgrades

To upgrade to the latest minor release of the DataHub version you are currently using:

1. Go to the Downloads page of the DataHub website.
2. Check to see if there is a more recent release of your version of the DataHub.
3. If so, download it and install it over your existing installation of the DataHub. The new version will automatically pick up any existing licenses and configuration.

Major Version Upgrades

To upgrade to a major new release of the DataHub, you need to first contact Cogent to obtain an upgrade license. Then download the latest archive of the DataHub and follow one of these two procedures:
Upgrade an existing DataHub installation

1. Install the new DataHub archive over your existing software.
   The installer will automatically detect your existing license and configuration. Since your license is not yet valid on this new version, it will not be used, and the DataHub will run in one-hour demo mode, with all features available.
2. Select the Licenses option. To see your existing license, check the Show unused license keys box.
3. Install the upgrade license you received from Cogent. The license number of the upgrade license must match the original license. If you have multiple upgrade licenses for several add-on features, install each one, ensuring that it matches its original.

   -winCDHOTv70:00075645:n:1:0:0:0:219d...

   -upg_winCDHOTv70_winCDHOTv90:00075645:n:1:0:0:0:236d...

   The End User License Agreement stipulates that each DataHub license may be used on only one machine at a time. So you may not use the original license on any other DataHub installation.

Upgrade and move to a new computer

Often part of the upgrade process is to move the DataHub to a new computer. Please see the section called “Backing Up or Moving a DataHub Installation” for information on moving configuration and license files.

1. If you want to use your existing configuration, copy and move it.
2. Install the new DataHub archive. (You do not need to install the earlier version of the DataHub.)
3. Continue with step 2 in Upgrade an existing DataHub installation (above).

Installing Version 9.x over previous versions

Please refer to Installing Cogent DataHub v9 on a system running an earlier version of the DataHub on the Cogent DataHub website.

Downgrading Version 9.x to a previous version

When DataHub Version 9 first runs, it makes a copy of the configuration folder from your previous version, and then upgrades the configuration to be compatible with V9. Your previous configuration is stored in the same folder, with an extension of `.bak`, where # is a numeric sequence number. For example, if your DataHub configuration folder is:

C:\Users\MyName\AppData\Roaming\Cogent DataHub

Then the configuration backup folder will be:

C:\Users\MyName\AppData\Roaming\Cogent DataHub.1.bak
When you revert to a previous version of the DataHub, you need to manually restore this backup to its original name. The full process is:

1. Stop DataHub and all tools, like WebView, Remote Config, DataPid, etc.
3. Install the older DataHub version, but do not start it.
4. Using the Windows file explorer, rename or delete the configuration folder (e.g., C:\Users\MyName\AppData\Roaming\Cogent DataHub)
5. Using the Windows file explorer, copy the backup folder to the configuration folder name. For example, copy:
   
   C:\Users\MyName\AppData\Roaming\Cogent DataHub.1.bak
   to
   
   C:\Users\MyName\AppData\Roaming\Cogent DataHub
6. Start the DataHub

**Using DataHub WebView after Downgrading**

If you are using WebView in Silverlight, you will need to clear your Internet Explorer browser cache so that Internet Explorer will load the older version of WebView. If you do not clear the browser cache you may see an error message like this:

![DataHub WebView](image)

**Configuration Files**

The Cogent DataHub uses multiple configuration files. When the program is first started it creates its primary configuration file, Cogent DataHub.cfg, along with other .cfg files that correspond to various DataHub plug-ins. These files get edited automatically when you make changes to features in the Properties window. When you shut down and restart the DataHub, it reads and uses the last saved configuration from each file.

There are two syntaxes used in DataHub configuration files. **DataHub Command syntax** is used in some, and XML syntax is used in others, depending on the complexity of the corresponding plug-in, and its implementation. Occasionally an experienced user might have reason to edit a configuration file directly, but in general modifications should be made through the Properties window.

DataHub configuration files are stored as UTF-8 text, without a byte-order marker (BOM). If you do choose to edit a DataHub configuration file, ensure that your text editor is storing the file in this format. The presence of BOM characters will cause the configuration
file to fail. Also, if you choose to edit a DataHub configuration file, whether by hand or through a text generation program, it is good practice to stop the DataHub before modifying the file.

**Configuration and License Files Location**

The Cogent DataHub stores its configuration and license (licenses.lic) files in the current user's directory, in a subdirectory named Cogent DataHub. You can find its configuration path in the DataHub About panel, by clicking the About button in the DataHub Properties window. Here are the typical locations:

*C: \Users\UserName\AppData\Roaming\Cogent DataHub*

- For Windows XP and Windows Server 2003:
  *

- For Windows Vista, Windows 7, 8, or 10, or Windows Server 2008, 2012, or 2016:

  If you ever need to back up or move the DataHub configuration, please refer to the section called “Backing Up or Moving a DataHub Installation”.

The Application Data or AppData directory might be a hidden directory.

If there is no private configuration when the DataHub starts, it will search for configuration files and license files from previous versions in the application installation directory, and copy them to the user's private configuration. If there are no old configuration files, the DataHub will copy all the current configuration files from the application installation directory. Thereafter, changes to the Cogent DataHub's properties (made through the Properties window) will only change the user's private configuration.

You can modify this behaviour in two ways with the following command line options:

1. Provide the `-H home` option which will indicate to the DataHub that it should store the private configuration and license files in the directory specified by `home`, rather than in a subdirectory of Application Data. The Cogent DataHub will still search for previous and default configuration files and licenses in the application installation directory as described above.

2. Provide the `-U` option which will indicate to the DataHub that it should not create private configuration files for each user, but store its configuration in the application installation directory.

If both `-H` and `-U` are specified, the `-U` flag is ignored.

You can also set the configuration folder globally using the DataHub Service Manager, even if the DataHub is not installed as a service. Here's how:
1. Open the DataHub Service Manager.
2. Click the button **Set Global Configuration Folder**.
3. Type or browse to the folder you want to use for configuration, and click **OK**.
4. Close the DataHub Service Manager window.
5. Start the DataHub normally.

This should result in the DataHub using the specified folder as its configuration folder whenever it is started without the `-H` option.

**Custom Configuration Files**

When the Cogent DataHub starts up, you may wish to have certain points and data structures get created immediately. To do this, you can create one or more custom configuration files. These files must be listed in the bottom of the **Scripting** option of the Properties Window.

Creating and editing custom configuration files should only be attempted by experienced users. If you do create a custom configuration file, we strongly recommend that it only be used for creating points and data structures, and not for standard configuration commands, such as those that are created and modified through entries in the Properties window. Doing otherwise could result in irregular behavior in the Cogent DataHub.

The following sets of commands are the only ones that should be used in a custom configuration file:

**General commands allowed in config files**

<table>
<thead>
<tr>
<th>create</th>
<th>mult</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>div</td>
</tr>
<tr>
<td>cset</td>
<td>lock</td>
</tr>
<tr>
<td>write</td>
<td>quality</td>
</tr>
<tr>
<td>cwrite</td>
<td>secure</td>
</tr>
<tr>
<td>force</td>
<td>append</td>
</tr>
<tr>
<td>cforce</td>
<td>dump</td>
</tr>
<tr>
<td>add</td>
<td>include</td>
</tr>
</tbody>
</table>

**Model-related commands allowed in config files**

<table>
<thead>
<tr>
<th>alias</th>
<th>private_attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>assembly</td>
<td>property</td>
</tr>
<tr>
<td>attribute</td>
<td>subassembly</td>
</tr>
<tr>
<td>defaultprop</td>
<td>type</td>
</tr>
</tbody>
</table>
Installation

instance

Only allowed in Windows config files

execute_plugin  show_debug_messages
load_plugin      show_event_log
unload_plugin    show_icon
load_scripts     show_properties
show_data        show_script_log

Custom configuration files can be created using a text editor like Notepad, and are written in Lisp syntax. They should be put in the same directory as the Cogent DataHub executable, such as C:\Program Files (x86)\Cogent\Cogent DataHub\ (it may be different for your installation). Each entry of the file contains either a command or a comment. Comments are marked with a semicolon character (;) at the beginning of each line.

For more information about commands and their syntax, please refer to Cogent DataHub Command Set.

A small custom configuration file might look like this:

```lisp
;;; Create some points in the default data domain and assign values.
(cset default:Point1 5)
(cset default:Point2 67.234)
(cset default:Point3 Hello)

;;; Create a new data domain
(create_domain NewDomain)

;;; Create some points in the new data domain and assign values.
(cset NewDomain:Point1 "A string")
(cset NewDomain:Point2 95)
(cset NewDomain:Point3 3.1519)
```

Back Up or Moving a DataHub Installation

It is possible to back up a Cogent DataHub installation, either for archival purposes, or for moving it to a different machine.

The Cogent DataHub End User License Agreement prohibits installing any DataHub license on more than one machine at a time. So, if you are moving the DataHub from one machine to another, after copying the license file from the first machine, you must delete it from that machine before installing it on the second machine.

To back up the necessary DataHub files, you will need to copy everything in the following directories:
• For DataHub version 9:
  C:\Users\UserName\AppData\Roaming\Cogent DataHub\n
  All custom scripts and configuration files must be copied from their directory on the first machine to an identical directory on the second machine.

  C:\Users\UserName\AppData\Roaming\Cogent DataHub\n  C:\Program Files (x86)\Cogent\Cogent DataHub\scripts\n  C:\Program Files (x86)\Cogent\Cogent DataHub\Plugin\WebServer\html\n
• For DataHub versions 7 and 8 running in Windows XP and Windows Server 2003:
  C:\Documents and Settings\User Name\Application Data\Cogent DataHub\n  C:\Program Files\Cogent\Cogent DataHub\scripts\n  C:\Program Files\Cogent\Cogent DataHub\Plugin\WebServer\html\n
In addition to this, if you have created any custom configuration files, they should also be copied from their directory on the first machine to an identical directory on the second machine.

**Known Issues**

Please refer to Known Issues and Breaking Changes on the Cogent DataHub website.
Getting Started

Running the Cogent DataHub

To run the DataHub, select the program using the Windows Start menu, or double click the desktop icon. Once started, the DataHub opens the Properties window:

When it starts, the DataHub puts an icon in the system tray. If you right-click on the icon, you will get a menu with several options, explained below:
Touch-screen users: press the DataHub icon for about one second. When you release, the pop-up menu will appear.
Script Log

QuickTrend

The pop-up menu also lets you **exit** the DataHub.

It is possible to run the DataHub with command-line options. Please refer to *Appendix A, Command Line Options* for more information.

**Pop-up Help**

You can get pop-up help in many parts of the Properties window by right-clicking the mouse over buttons or text.

**Exit**

You can terminate the DataHub by right-clicking the Cogent DataHub icon in the system tray, and selecting **Exit** from the pop-up menu. After a few seconds the icon should disappear, indicating that the DataHub has terminated.
You must explicitly exit the DataHub to terminate it. Otherwise it continues to run in the background even if you close the Properties, Data Browser, and Event Log windows.

Test with simulated data

There is a data-generating program that comes with the Cogent DataHub called DataSim. You can run DataSim locally to create data for various connection scenarios.

1. Start the Cogent DataHub if it isn’t already running.
2. Start DataSim using the Windows Start menu, or by double clicking the desktop icon.
3. Open the DataHub Data Browser by right clicking on the DataHub system-tray icon and choosing View Data from the pop-up menu.
4. Select the DataSim data domain in the left-hand pane of the window.
   The Data Browser window should fill with simulated data updating in real time. The four updating data points are Ramp, Sine, Square, and Triangle.
5. You can also click the View Connections button near the bottom of the Properties window to see your currently configured connections:
Getting Started

The Connection Viewer shows all active connections in the Cogent DataHub.

6. You can click the More... button in DataSim to access some options for changing the data feed.

Briefly, you can change the Configurable Options and click the Apply Changes button to apply them. The Waveform Parameters and Update Frequency are all points in the Cogent DataHub, and the corresponding points change their values in the Data Browser as you make the changes. Changing the Data Domain from DataSim will yield no results until custom data domains are configured for the DataHub.

If you shut down DataSim, its points will still appear in the DataHub until it is shut down and restarted. Please refer to the section called “Data Points” for more information on creating and deleting points.

Now you are ready to start using the DataHub.

Connect to an OPC DA server

To connect to an OPC DA server, you need to configure the Cogent DataHub to act as an OPC Classic client. Here’s how:

YouTube Click here to watch a video.
1. Right click on the Cogent DataHub system-tray icon and choose Properties.
2. In the Properties window, select OPC DA.

![OPC Classic Data Access (DA) Configuration]

3. Check the Act as an OPC Client box. Since the DataHub can be a client to more than one OPC server, you need to specify which OPC DA server you are going to connect to. To add a server, click the Add button and fill in the fields in the Define OPC Server Window:

![Define OPC Server]

4. Type in or select the necessary information as appropriate.
a. The first four fields define the OPC server:

- **Connection Name** Type a name to identify this connection. There should be no spaces in the name. It doesn't matter what name is chosen, but it should be unique to other connection names.
- **Computer Name** Type in or select from the drop-down list the name or IP address of the computer running the OPC server you want to connect to.
- **OPC Server Name** Select the name of the OPC server that you are connecting to from the list of available servers.
- **Data Domain Name** Type the name of the DataHub data domain in which the data points will appear.

b. You can specify how the data is to be transferred.

- **Maximum update rate (milliseconds)** Enter the maximum rate you wish the data to be updated. This is useful for slowing down the rate of incoming data. The default is 0, which causes values to be updated as soon as possible. This value is also the polling time used by asynchronous and synchronous reads (see below).
- **Read Method** Choose how to read data from the OPC server:
  - **Asynchronous Advise** The OPC server sends a configured point's data to the DataHub immediately whenever the point changes value. This is the most efficient option, and has the least latency.
  - **Asynchronous Read** The DataHub polls the OPC server for all configured points on a timed interval (set by the **Maximum update rate**). This option is less efficient than Asynchronous Advise, and has higher latency.
  - **Synchronous Cache Read** The DataHub polls the OPC server for all configured points on a timed interval (set by the **Maximum update rate**), and this thread waits for a reply. This option is less efficient than Asynchronous Advise or Read, and has higher latency than either of them.
  - **Synchronous Device Read** The DataHub polls the PLC or other hardware device connected to the OPC server for all configured points on a timed inter-
val (set by the **Maximum update rate**), and this thread waits for a reply. This is the least efficient of all of these options, and has the highest latency.

- **Write Method** Choose how to write data to the OPC server:
  - **Asynchronous Write** provides higher performance. The Cogent DataHub writes changes in point values to the OPC server without waiting for a response.
  - **Synchronous Write** elicits a quicker response from the OPC server, but results in lower overall performance. The Cogent DataHub writes changes in point values to the OPC server without waiting for a response. This option is useful if the OPC server doesn't support asynchronous writes at all, or if it can't handle a large number of them.

Depending on the OPC server you are configuring, you might have an option to use OPC DA 2.0 or 3.0. Please refer to the **Data Transfer** explanation in the OPC section of the Properties Window chapter for more information.

c. There are several optional entries:

- **Treat OPC item properties as DataHub points** lets you register and use non-standard OPC item properties as points in the DataHub. Generally you won't need this unless you plan to use the DataHub to distribute changes to values of the non-standard properties on your OPC items.

  The Cogent DataHub will monitor these properties only if the OPC server exposes them as OPC items. If the properties do not show up when using this check-box, this means that the server does not expose the non-standard properties as items.

  Some OPC DA servers are slow to register their OPC items and properties. Using this option with one of these servers can significantly slow the start-up time of the DataHub

- **Read only: Mark all items as Read-Only** lets you specify that the OPC server be read-only, regardless of how individual items are specified. Items in the DataHub that originate from such an OPC server will be read-only to all DataHub clients.
• **Replace item time stamps with local clock time** allows you to set the time-stamps for the items from this server to local clock time.

• **Force connection to use OPC DA 3.0** lets you choose the DA 3.0 write methods from the **Write Method** drop-down box. It will also instruct the Cogent DataHub to attempt to browse the server using DA 3.0 browsing. This setting will override any automatic information that the Cogent DataHub may determine about the server based on the server's registry entries.

• **Never use OPC DA 3.0** removes the DA 3.0 write methods from the **Write Method** drop-down box, and will instruct the Cogent DataHub to only use DA 2.0 browsing. This setting will override any automatic information that the Cogent DataHub may determine about the server based on the server's registry entries.

For more information about OPC DA 2.0 and 3.0, please refer to the Data Transfer explanation in the OPC section of the Properties Window chapter.

• **Set failed incoming values to zero** The OPC DA spec requires an OPC DA server to send an **EMPTY** (zero) value whenever it sends a failure code in response to an item change or a read request. Some OPC servers, however, send a valid value with the failure code under certain circumstances. To ignore any such value from the OPC server and assume **EMPTY**, keep this box checked (the default). If instead you want to use the value supplied by your OPC server, uncheck this box.

  Unchecking this box will make the Cogent DataHub's behavior non-compliant with the OPC specification.

• **Never use OPC DA 2.0 BROWSE_TO function** disallows the **BROWSE_TO** function when communicating with OPC DA 2 servers. Sometimes an OPC server will have problems with this function that prevent the Cogent DataHub from connecting to it. Checking this box might allow the connection to be established in those cases.

• **Never attach to an in-process COM server** Most vendors include both an in-process and out-of-process COM server with their OPC DA server installation. If both options are available, the DataHub connects to the in-process server, as it is generally the better choice. This option forces the DataHub to consider only out-of-process servers.

  Why is this useful? An in-process server is implemented as a DLL that is loaded into the client's address space. This makes the client very dependent on the good implementation of the server. If there is a crash in an in-process server, the client also crashes. An out-of-process server is implemented as a separate executable. The client communicates with an out-of-process server using the inter-process communication mechanisms in DCOM. In theory an in-process server will be faster than an out-of-process server, but sometimes the in-process server is less robust than the out-of-process server and leads to in-
stability or malfunction in the client.

- **Allow VT_EMPTY canonical type for OPC DA2** The VT_EMPTY canonical type may be incompatible for a particular combination of OPC server and client. For example, some clients or servers that were built before 64-bit integers were common may fail when presented with a 64-bit number. These options (DA2 and DA3) allow you to enable or disable the VT_EMPTY canonical type, either for trouble-shooting or as a permanent part of your configuration.

- **Allow VT_EMPTY canonical type for OPC DA3** See above.

- **Wait for server running state** Every OPC DA server takes a little time to initialize before it will allow client connections. This option lets the user specify the time to wait for the OPC server to initialize. The wait time is a maximum; if a server initializes before this time, the DataHub will connect right away. If the server doesn't initialize within this time, the DataHub will report this in the Event Log, and then try to connect anyway.

- **Pause before reading data** specifies a time for the DataHub to pause before reading the OPC server's data set. Some OPC DA servers report that they are running, but have not yet received the full data set from the process. If the DataHub attempts to connect right away, it might get a partial data set. The pause is fixed; it will always last for the full time specified.

  The two above times are added together. The DataHub will wait until the server is initialized (or until the specified "wait" period is complete) and then pause for the specified "pause" time, before trying to read data from the server. For example, with the defaults of 5000 and 1000, at least 1 second and at most 6 seconds will elapse before the DataHub tries to read the data set.

  
  d. Finally, you can specify how the OPC items get selected. You can select them manually or load all of them.

  
  ![Item Selection](image)

  **Manually Select Items**

  Check the **Manually Select Items** box and press the **Configure Items** button to open the OPC Item Selection window, where you can specify exactly which
points you wish to use:

You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. Follow these guidelines for making selections:

- To select a server item from the right-hand pane, click its check-box.
- To highlight a list of consecutive server items, click the first item, hold down the Shift key, and then click the last item. To highlight separate server items, hold down the Ctrl key as you click each item. To select a group of highlighted items, use the Spacebar.

These may not function as described for Windows NT or Windows 2000 operating systems.

- Selecting a server item does not automatically add any of its child items. Each child item must be added separately. To view child items, click the + sign in front of the item. If an item has one or more children that have been selected, the item name(s) will appear in bold.

- To delete selected items from the right-hand pane, highlight them and press the Remove Selected button. Use the Shift and Ctrl keys as above to highlight groups of selected items.

You may also configure dynamic items on the server. As you type in the Server Item ID, the system will fill in an identical DataHub Point Name for you (which you can change at any time). Press the Enter key or the Apply button to create the item. Checking the Copy names from selection box will fill in
the entry with the name you select from the Selected Items list (above). The Recognize branch delimiter in point name option lets you select and apply a point delimiter for your dynamic items.

**Load All Items on Server**

In addition to manually loading items, you have the option in the Define OPC Server dialog to register all points, or filter for groups of points, from the OPC server.

In the **Server specific item filters** you have the option create filters to select partial data sets. If you don’t enter anything here, the DataHub will query the OPC server for all of its items and register them. The filters are all applied on a logical ‘OR’ basis, i.e. if a point satisfies the condition of any filter, it gets registered with the DataHub.

- Click the Add... button to add a filter. The Edit a filter string window will appear:

Enter a string or a pattern to match one or more item names in the OPC server. Each server has its own syntax for pattern matching, so you may have to experiment a little to get exactly the points you need. Commonly, the symbol * matches any number of characters, while the symbol ? often matches a single character. In that case, an entry of ?a* would bring in all items with a as the second letter in their names.

- Click the Edit... button to open the Edit a filter string window and edit an existing filter. You can do the same thing by double-clicking a filter string in the list.

- Click the Remove button to remove a selected filter from the list.

5. Click the Apply button in the Properties Window. The DataHub should begin to act as a client to the OPC server. Messages will appear in the **Status** column indicating the status of the connection:

Configuring After you click the OK button in the Define OPC Server dialog until you
click the **Apply** button in the Properties window.

**Server Lookup** The DataHub is looking for the OPC server.

**Server Attach** The DataHub has found the OPC server and is connecting. It may be waiting for the server running state, as explained previously.

**Pause \text{nnnn} ms** The DataHub is paused before reading data, as explained previously.

**Running** The DataHub is connected to the OPC server and exchanging data.

**Disconnected** The DataHub has disconnected from the OPC server.

You can verify the connection using the **Data Browser** or the **Connection Viewer**. You can change server settings at any time. The Cogent DataHub will reconnect and apply the changes when you click the **Apply** button in the Properties Window.

### Connect from an OPC DA client

When you start an OPC DA client it should immediately connect to the Cogent DataHub, because the DataHub is preconfigured to act as an OPC DA server. If the DataHub is not running, the OPC client will attempt to start it.

If your client does not connect, you can check the DataHub configuration as follows:

1. Right click on the DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **OPC DA**.
3. Ensure that the **Act as an OPC Server** box is checked.

   ![Configure OPC DA settings](image)

   If your OPC client requires that you hand-enter the OPC server name, use either `Cogent.CogentDataHub` or `Cogent.CogentDataHub.1`.

4. For information on any of the other options, please refer to the **OPC DA Server** section in Properties.
5. Click **Apply** button at the bottom of the Properties window to apply the change. You can view connections with the **Connection Viewer**.
Test MQTT

Here is a quick way to test both the DataHub MQTT Broker and MQTT Client, by connecting them to each other.

Configure the MQTT Broker

1. In the Properties window, select MQTT Broker.

2. In the Point Names options, check the Place all points in this data domain box, and enter MQTTBroker for the domain name.

3. Click Apply.

The DataHub’s MQTT Broker is now ready to accept MQTT client connections for the MQTTBroker domain. For more information about the MQTT Broker feature, please refer to the section called “MQTT Broker”.

Configure the MQTT Client

1. In the Properties window, select MQTT Client.

2. Click the Add button to open the Connect to MQTT Broker window:
3. In the **Standard MQTT** section, **Connection** tab, enter the following:
   - **Label:** TestMQTT
   - **Host:** localhost
   - **Port:** 1883 (the default)
   - **Keepalive:** 30 (the default)
   - **Retry rate:** 5000 (the default)
   - **Maximum update rate:** 0 (the default).

4. In the **Push data points to the MQTT broker** section, **Available Points** list, open the **DataPid** tree and select the point **Mv**. (If you don't see the **DataPid** domain, start **DataPid**.)

5. Click the **MQTTBroker** domain in the domain list to highlight it.
6. Click **OK**, and then **Apply**.
7. Click the **View Data** button in the Properties window to open the Data Browser. In the **MQTTBroker** domain you should see the value for **DataPid.Mv** updating.
You have now configured the DataHub MQTT Client to send a value to the DataHub via the MQTT Broker. For more information about the MQTT Client feature, please refer to the section called “MQTT Client”.

Connect to remote data

There is a DataHub running on a Skkynet cloud server that you can connect to for testing. To configure the Cogent DataHub to receive that test data, just follow these steps:

1. Right click on the Cogent DataHub system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror.
3. In the Tunnelling/Mirror Slave section, check the Act as a tunnelling/mirroring slave to these masters box.
4. Click the Add Master... button to open the Tunnel/Mirror Master Configuration window:
5. Fill in these entry fields as follows:
   - **Primary Host** demo.skkynet.com
   - **Port** will be set automatically by the system, 80 for WebSocket and 443 for Secure (SSL) (see below).
   - **Local data domain** cloud
   - **Remote data domain** DataPid
   - **Remote user name** demo/guest
   - **Remote password** guest
   - **WebSocket** must be selected.
   - **Secure (SSL)** is optional.
   There is no need to make or change any other entries.

6. Click **OK** to close the Tunnel/Mirror Master window, and then click **Apply** in the Properties Window.

7. Open the **Data Browser** and click the **cloud** and the **PID1** data domain name in the left-hand pane of the window.

   The Data Browser window should show some data updating in real time. Any delays in updates to these points are due to slow network speed or high traffic volumes.
You can also use the Connection Viewer to see all active connections in the Cogent DataHub.

For more information on tunnelling/mirroring, please refer to the section called “Tunnel/Mirror”.

Custom Connections

There are several ways to make custom connections to the DataHub from Windows, Linux and other programs, as well as web browsers and embedded devices, using TCP, SSL, and in some cases, WebSocket protocols.

- For Windows programs see DataHub APIs for C++, Java, and .NET
- For Linux programs see Sending Commands by TCP
- For other programs see Sending Commands by TCP
- For embedded devices see Embedded Toolkit (ETK)
- For web browsers, using Javascript, contact Cogent

Performance Limitations

Data points and update rates

There are no hard limits on the number of data points or the update rates for the data in the DataHub. More data points require more memory, and more updates require more CPU. Different protocols require different amounts of CPU and memory.
Total throughput

The total throughput of the DataHub will depend on the total load, not just the input rate. If the DataHub is receiving 10,000 point changes per second, and there is 1 client connection, that will count as 10,000 point changes per second. If there are 5 client connections receiving that data, that becomes effectively 50,000 total changes per second being processed by the DataHub.

Data Domains

There are no hard limits on the number of data items or data updates per domain.

Event Log

Configuring the DataHub Event Log to log to disk has a big impact on performance. If you are seeing slow performance, try disabling logging to disk.

Specific protocols and features

OPC DA and UA

There are no hard limits on the number of OPC tags or values that can be sent or received by OPC servers or clients. The number of data points and throughput is limited by memory and CPU. In test situations for OPC DA on reasonably modern hardware we have achieved over 100,000 data point changes per second. The OPC client and server code in the DataHub automatically combines multiple point changes into a single OPC message whenever it can. Generally, OPC DA is faster than OPC UA on the same machine, and DHTP (DataHub tunnelling) is the fastest way to network OPC data. OPC UA requires much more memory than other protocols.

Redundancy

There are no hard limits on the number of points for the Redundancy feature. It is limited by memory and CPU. The total data point count is the sum of the data points counts in each domain. The number of domains is the total of all input and output domains for all redundant connections. For example, a single redundant configuration has 2 inputs and 1 output, for a total of 3, requiring enough memory for 3 times the number of points as are in one of the source domains.

Bridging

There are no hard limits on the number of points that can be bridged. The Bridging feature is built directly into the data path of the point change-handling code of the DataHub, so it is as efficient as a normal point change. CPU load is based on point changes, while the memory load is based on the number of points, plus a small amount for the Bridging configuration itself.
ODBC

Our testing of the Database Write (logging) feature on MS-SQL and MySQL shows that typically the database server can handle up to about 1,000 transactions per second on a reasonably fast computer. The limiting factor is the speed of the database server. We have tested the DataHub with Times Ten database, which is faster than a typical SQL database. On that, the DataHub can send about 10,000 transactions per second. The limiting factor in that case is the speed of the DataHub.

OPC A&E

OPC A&E conditions contain many individual properties. When you select the option to Make A&E Status available as individual data points, it will result in multiplying the number of A&E conditions by 50 or more. So, an A&E server with 1000 conditions could multiply to 50,000 data points or more when that option is selected.

DHTP (Tunnel) Bandwidth

The per-point transmission size for DHTP is approximately 60 bytes plus the length of the data point name. The tag name is UTF-8 encoded, so for names that can be represented in the 7-bit ASCII character set, that is 1 byte per character. Where possible the DataHub combines point transmissions into single TCP/IP packets to reduce TCP header overhead, up to 1 kB in size.

The DataHub does not transmit all points. It transmits only those that have changed since the previous scan. Consequently, the size of an OPC scan group does not matter. The important calculation is how many points within the scan group change with each scan. You can see the point transmission rate in the "View Connections" window in the DataHub.

If the data transmission is write-only then there is no acknowledgement back from the server. If the data transmission is read/write then the server will send back an equivalent message for every point change, effectively doubling the bandwidth requirement.

There is some overhead in SSL when the connection is established—typically around 10 kB. Once the connection is established there is about a 2% increase in bandwidth. The biggest overhead is when establishing the connection, not during data transmission.

DHTP (Tunnel) Buffering

Tunneling takes advantage of the TCP/IP buffer to smooth changes in network speed. The buffer is set to 8 kB, which roughly translates into about 100 data point messages. If this buffer fills, you will see that manifest in the DataHub's Connection Viewer as an increment to the "Blocked" counter for the connection. If a data point value changes while the connection is blocked, the oldest value for that point is dropped, and the Dropped counter in the connection viewer will increment. If the Dropped counter remains at zero then all data point changes are successfully being transmitted through the tunnel. If the Blocked counter is increasing then the average data rate is faster than the network can handle.
Reporting Problems

The DataHub is designed to be easy to use, but sometimes people run into problems, and need technical support. Many of these problems can be solved with Cogent's online resources, but sometimes you might need some assistance from Cogent.

Before submitting a report, there are several things you can quickly check:

• Are you running the latest version of the DataHub?
• Is the DataHub installed and configured correctly?
• Does it run OK?
• If you are tunnelling, is your connection to the network and/or Internet up and functioning normally? Do your firewall settings allow the necessary access? Can you ping between machines?

If all of those check out and you are still seeing the problem, then you'll need to collect the required information before taking action. For effective trouble-shooting, everyone involved will need the best possible understanding of the problem. This will include a clear description of the system and what you are seeing, as well as output from DataHub logs. Screenshots and DataHub configuration files may also be helpful.

System and Problem Descriptions

The system description should include all data connectivity information:

• Data communication protocols used (OPC, Modbus, etc.)
• The number of DataHubs in use
• DataHub features in use at each node
• The DataHub version number
• A system diagram, if relevant

The problem description should contain as much detail as possible, such as:

• When was the problem first noticed?
• What kind of action triggers it?
• What has changed in the system recently?
• How often does it happen?
• How to reproduce it?
• Any other useful information.

DataHub Logs

There is often useful information in the DataHub's Event Log. If you need help understanding the Event Log output, you can contact Cogent. You can copy the text of the log...
right from the display window, if the whole problem is visible. Otherwise, we recommend selecting Log to file and zip up and send the whole file.

Another useful DataHub log is the Script Log. This records any errors in DataHub scripts. Also, the Database feature (ODBC) uses DataHub scripting, so any errors connecting to a database will appear in the Script Log as well. For help understanding the Script Log, you can contact Cogent.

Screenshots

Screenshots of the problem can be helpful, especially for seeing connected programs like OPC servers or database tables. Of course, the Event Log output and DataHub configuration files usually have the most complete information about the DataHub itself.
Configuration Directory

Sometimes Cogent will need a copy of the DataHub configuration files. These are all in one directory, located here:

For Windows XP and Windows Server 2003:

C:\Documents and Settings\UserName\Application Data\Cogent DataHub


C:\Users\Username\AppData\Roaming\Cogent DataHub

At Cogent's request, the directory can be zipped up and emailed to us.

Crash Dumps

Crash dumps are frequently helpful in narrowing down the cause of a crash, allowing us to try to reproduce the problem. Windows can create crash dumps in one of the following locations:

C:\Windows\Minidump
C:\Windows\CrashDumps
C:\Users\Username\AppData\Local\CrashDumps
C:\ProgramData\Microsoft\Windows\WER\ReportQueue

A full crash dump file will likely be 200 MB or more in size, have the word CogentDataHub in its name somewhere, and have a .dmp suffix.

If you can find a dump, please ZIP it and send it to Cogent, or put it on a file sharing system where we can download it.

If there is no crash dump, you can set a registry entry to cause Windows to create one on the next crash by setting DWORD - HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows Error Reporting\LocalDumps\DumpType to 2. If this registry value does not exist then you will need to create it.

Final Check

Before contacting Cogent, make one final check to see if you might be able to solve the problem yourself.

1. Review the relevant parts of the documentation, available online or by clicking the Help button in the DataHub Properties window.
2. Double-check for obvious configuration errors.
3. Upgrade the DataHub to the latest version, if possible.
4. Look at the Event Log output, and Script Log (if applicable) for when the problem oc-
You are now ready to report the problem.

**Contact Cogent**

If these steps or any other approaches do not resolve the issue, then you should contact Cogent as soon as possible. Please send along all of the information that you have gathered:

- Complete descriptions of the system and the problem, including screenshots, if any.
- Output from the Event Log and/or Script Log.
- The DataHub configuration directory.
- Any crash dumps, if relevant.
- If it becomes necessary, would you be able to make your system available for desktop sharing? Please let us know. We prefer WebEx, but can also use TeamViewer if necessary.

We try to respond to all support requests promptly.
Properties Window

This is where you can configure the Cogent DataHub.

For All Options

For all the options (General, OPC, Tunnel, Bridging, etc.) in this window:

- The **Interface Language** list lets you choose a language for the interface. If you don't find your language, you can contact Cogent for instructions on how to add a translation to the DataHub source code.
- The **View Data** button starts Data Browser.
- The **View Connections** button starts Connection Viewer.
- The **Event Log** button starts Event Log.
- The **Script Log** button starts Script Log.
- The **About** button provides some general information about the software.
• The **OK** button applies changes and closes the window.
• The **Apply** button applies changes but leaves the window open.
• The **Cancel** button closes the window without applying any changes.
• The **Help** button opens the help window for the current option.

**General**

This first option in the Properties Window lets you control how the DataHub starts up, and how changes to data are transmitted.

**Data Changes**

**Do not transmit insignificant changes** will reduce traffic by allowing only significant changes to the data to be sent. A change is *significant* if a property of the point other than the time-stamp changes. That normally means a change to either the value or the quality of the point. A change in only the time-stamp is considered insignificant. Some polled data sources change the time-stamp on each cycle, even if the value doesn't change. If network bandwidth is a concern, you can use this option to update the point only when the value has changed.

**Do not transmit changes with an old timestamp** allows only current or future changes to be sent.

**Automatically add a timestamp to unstamped changes** stamps the current time onto any changes that haven't already been time stamped.

This should stay checked unless you have specific reason to uncheck it. Unchecking it may cause changes made through DataSim, the Data Browser, and other programs to receive timestamps of `0 (Dec 31 19:00:00:00.000)`. If this button is *unchecked* and the **Do not transmit change with an old time-stamp** is *checked*, then any changes with a 0 timestamp won't get transmitted at all.
Properties Window

Startup

Show the splash screen at startup lets you hide or show the startup screen with the DataHub image.

Show a warning message if running in demonstration mode lets you hide or show the message telling you the demo will terminate in one hour.

Start when you log on to Windows causes the DataHub to start up whenever you log on to Windows.

Allow only one running program instance prevents more than one Cogent DataHub from running at the same time.

Declare these data domains

In this area you can add, edit, or remove data domains for the Cogent DataHub. For more information about data domains, please refer to the section called “Data Domains”.

To add a data domain, click the Add button and fill in the name in the Data Domain Name Window:

To change a data domain, double-click it or select it and click the Edit button. To remove a data domain, highlight it and click the Remove button.

Checking the Automatically add data domains requested by clients box automatically adds a data domain whenever a client requests it. If for some reason you want to limit the data domains to those listed, you should make sure this box is not checked.

OPC UA

The OPC UA option lets you configure the Cogent DataHub to act as an OPC UA (Unified Architecture) server, an OPC UA client, or both simul-
For more information on OPC, please refer to the section called "OPC Protocol" and Appendix F, OPC Overview.

For step-by-step instructions to configure the DataHub as an OPC UA client or OPC UA server, along with more information about the DataHub's implementation of the OPC UA standard, please refer to OPC UA Connections.

Click here to watch a video.

OPC UA Client

The Cogent DataHub can act as a client to OPC UA servers.

Check the Act as an OPC UA Client to these servers box to activate OPC UA client functionality. Since the DataHub can be a client to more than one OPC UA server, you need to specify server information for each OPC UA client connection. Once you have a server listed, you can activate or deactivate the connection using its On check box.

To add a server, press the Add button to open the Configure OPC UA Data Access Server window described below. To edit a server, double-click it or select it and press the Edit button to open that window. To remove a server, highlight it and click the Remove button.

The Configure OPC UA Data Access Server Window

To define or change an OPC UA server connection, click the Add or Edit button to open the Define OPC Server Window:
Properties Window

Connection Name
A name used by the Cogent DataHub to identify the connection. It doesn't matter what name is chosen, but it should be unique to other connection names.

Discovery Domain
A list of UA Discovery Servers to which you can connect.

Endpoint URL
A list of available Endpoint URLs for the chosen Discovery Server. For each Discovery Domain, the system will attempt to list its endpoint(s), providing feedback as follows:

- Indicates that the endpoint discovery is in process.
- Indicates that the endpoint discovery has failed.
- Indicates that the endpoint discovery has succeeded.

If a connection has already been configured, and the Configure OPC UA Data Access Server Window is opened for editing, the Endpoint URL will first appear as previously configured. The DataHub will then attempt to validate the endpoint, with the status icon changing first to In Process (ıc), and then to either Failure (x) or Success (v).

If at any time you initiate a search by pressing the server refresh button ( kuruluş ), and the system fails to locate an endpoint URL, then it will leave the Endpoint URL entry field empty.

Security Policy
A list of available security policies available on the OPC UA server for this connection.

User Token Type
The log-in method used for this OPC UA server and session. Some possible options are:

- Anonymous
  The UA server allows any user to connect.
- UserName
  The UA server requires a user name and password.
- My Certificate
  The UA server allows you to use your DataHub's own certificate.
- Another Certificate
  The UA server requires a certificate other than your DataHub's own certificate.

User Identity
This will change depending on the User Token Type (above), allowing for the entry of a certificate file path or a user name and password, as appropriate.

Always accept invalid server certificate
Tells the client to always accept the server certificate, even if the certificate is invalid.
or if it changes in the future.

Selecting this option will disable server certificate verification for this connection, exposing the connection to man-in-the-middle attacks. Use with extreme caution.

**Continue to accept server certificate when it expires**

This option allows a UA certificate to be accepted outside of its valid time window, meaning that expired certificates can continue to be used. Checking this box also keeps the UA server and client connected if their system clocks ever get out of synch. And this option also supports connectivity for OPC UA clients running on embedded systems without system clocks, or whose system clocks cannot be adjusted.

If you are using the http protocol along with a security policy, then the clocks on the UA server and client machine must match within 5 minutes at all times. This is a requirement of the WCF subsystem that implements the HTTP security. If you are not able to synchronize the clocks on the server and client machines this closely, you should try either the opc:tcp or https protocol, which do not rely on WCF for the underlying security and so do not exhibit this problem.

**Do not disconnect when the server reports a failed state**

By default, if the server is in a non-Running state the DataHub disconnects and puts a message in the Event Log. Checking this box lets you override that behaviour and maintain the connection to the server.

**Connection Test**

To test the connection, click the Connection Test button. The system will open the Connection Test window, and you can watch as it checks the parameters, then creates a channel and session, and then activates the session.
If there is a problem at any point, the **Message** box will provide some trouble-shooting tips.

**Data Transfer**

There are several options for specifying how the data is to be transferred:

1. **Maximum update rate (milliseconds)**
   - This option lets you specify an update rate, useful for slowing down the rate of incoming data. The minimum value is 10. This value is also used as the polling time for asynchronous and synchronous reads (see below).

2. **Read Method**
   - **Subscription** The DataHub registers with the UA server for all configured points, to be received on an event-driven basis. Whenever a point value changes, the new value is sent immediately to the DataHub. This option is more efficient than **Synchronous Read** or **Asynchronous Read**, and has lower latency than either of them.
   - **Synchronous Read** The DataHub polls the UA server for all configured points on a timed interval (set by the **Maximum update rate**), and this thread waits for a reply. This option is less efficient than **Subscription** or **Asynchronous Read**, and has higher latency than either of them.
   - **Asynchronous Read** The DataHub polls the UA server for all configured points on a timed interval (set by the **Maximum update rate**), and does not wait for a reply. This option is less efficient than **Subscription**, and has higher latency.

3. **Write Method**
   - **Asynchronous Write** The DataHub writes to the UA server and does not wait for a response. This provides the highest overall performance.
   - **Synchronous Write** The DataHub writes to the UA server and waits for a response each time. This elicits a quicker response for a given item from the UA server, but results in lower overall performance. This option is useful if the UA server doesn't support asynchronous writes at all, or if it can't handle a large number of them.
Monitored Item Queue Size
The maximum number of items between polls that get stored on this server.

Maximum Request Item Count
The OPC UA spec allows a UA server to specify the number of items it will allow per request. Here you can adjust the DataHub default of 500 to what the server allows, if necessary.

Only transmit GOOD quality data to this server
Restricts point updates from the DataHub to the server to only those with "Good" quality.

Do not accept null data from this server
Restricts point updates from the server to the DataHub to only those with non-null values.

Create multiple subscriptions using Maximum Request Item Count
The Maximum Request Item Count (above) specifies the maximum number of nodes per subscription. With the Create multiple subscriptions... option checked (the default), the DataHub will use this number to decide the maximum number of nodes per subscription. However, if this number is small and the total number of nodes is large then the number of requested subscriptions could exceed the subscription count limit of the server. Unchecking this box will solve that problem by putting all of the nodes into a single subscription.

Advanced
OPC UA communication is governed by a number of timeout and length limits. Normally you do not need to adjust these, but in some cases you may need to extend timeouts for poorly behaved networks, or to reduce message length limits to accommodate servers with limited buffer sizes. Most commonly you would need this with resource-constrained embedded servers.

Clicking the Advanced button opens the Advanced Connection Settings window:

![Advanced Connection Settings](image)

Here you can enter the following Transport Quotas:
• **Operation Timeout**: If a network operation does not complete within this time, abandon the operation. This will normally cause the DataHub to drop the connection and re-attempt it after a few seconds.

• **Maximum String Length**: The longest permissible string, in bytes. UTF-8 strings may use up to 5 bytes per character.

• **Maximum Byte String Length**: The longest permissible byte string (data type `Byte String`), in bytes. Byte strings are uninterpreted sequences of bytes that may represent any data.

• **Maximum Array Length**: The maximum number of array members for any data value.

• **Maximum Message Length**: The longest permissible message, in bytes.

• **Maximum Buffer Size**: The maximum buffer size, in bytes. The buffer size determines how much data can be read in a single network read call and does not limit the maximum size of a message. You could use this setting to optimize memory usage or reduce the number of network reads in the DataHub.

• **Channel Lifetime**: The lifetime of the client channel, in milliseconds. This specifies how long the server will keep a broken channel around while waiting for a client to reconnect.

• **Security Token Lifetime**: The lifetime of a security token, in milliseconds. This specifies how long a security token can be used without renewal.

### Item Selection

You can select all nodes, select nodes manually, or both.

![Item Selection](image)

#### Load All Nodes on Server

With this option you can load all data nodes on the OPC UA server, or filter for groups of nodes.

When you choose this option, the DataHub is configured to provide all data nodes, but not the Server nodes. This is done as a convenience, because in most cases few, if any, Server nodes are needed. To additionally get Server nodes, you can select them manually.

#### Manually Select Nodes

Select the **Manually Select Nodes** option and press the **Configure Nodes** button. This opens the Configure Nodes window, where you can specify exactly which nodes you wish to use:
You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. To view sub-branch and leaf items, click the + sign in front of the item to show the children. You can select many items together like this:

1. Expand all of the branches containing points that you want to add.
2. Click the name of the first point (not the check box).
3. Go down to the last point, hold down the Shift key and click the name. All of the names should become highlighted.
4. Press the Space Bar.

That should select all of the highlighted points. It will not select nodes that are not visible.

Selecting just a branch by itself will not include any of its sub-branches or leaves, but selecting a leaf item will automatically include all of its branches.

Checking the box Select variable only data point will ensure that the only nodes you choose are data nodes.

A + in front of an item does not necessarily mean that the item has children. You must click the + sign to find out.

Data Domain Name
The name of the DataHub domain into which the data points will be placed.

OPC UA Server
The Cogent DataHub can act as a server to any number of OPC UA clients.
Any changes made here will restart the OPC UA server when you click the **Apply** button.

Check the **Act as an OPC UA Server** box and click the **Apply** button to have the Cogent DataHub function as an OPC server. You can choose one or more of the available protocols, modify the default selection using the **Advanced** option (explained below), or change the port number by double-clicking or using the **Edit Port...** button. You can also use the **Copy Endpoint to Clipboard** button to make a copy of this server's endpoint.

**Computer Name/IP**

The host name or IP address of the computer on which the DataHub is running. This will be integrated into the server URL visible to a connecting client. The default is the host name.

**Endpoint Name**

The endpoint name that will be integrated into the server URL visible to a connecting client. The default is `CogentDataHub/DataAccessServer`.

Some UA clients cannot connect to a UA server unless the server name is left blank. For these cases, the DataHub can be configured with a blank server name as follows:

1. Clear the **Endpoint Name** entry field so that it is blank.
2. Uncheck the **HTTP** and **HTTPS** protocols, as these are not supported when the **Endpoint Name** is blank.
3. Click **Apply** to save the changes.

The DataHub UA server will restart with a blank user name, allowing a UA client to connect to it using a simple Endpoint URL, for example:

```
opc.tcp://192.168.1.1:52310/
```

Some UA clients may require some or all of the following information about the DataHub OPC UA server:

- **Namespace** http://www.cogentdatahub.com/DataHub
- **Namespace ID** 2
- **ID type** This information should not be exposed to the user.
• **ID** Something like this: `ns=2;s=DataPid:PID1.Mv`. Generally the syntax is `ns=2;pointname`. The namespace is always 2.

• **Type** Typically the canonical type of the node (ID above) retrieved from the server through a client request.

• **Access to data point** The client application developer will need to provide this information, such as read-only or read-write.

**Advanced**

The default configuration covers most typical client connection requirements. If you need to modify these, you can click the **Advanced** button.

**Advanced**

Clicking the **Advanced** button opens the UA Server Properties window:

![UA Server Properties](image)

This window allows you to configure the following options for how the DataHub functions as a UA server: Some of these settings will require you to restart the OPC server, others will not, as indicated.

**General**

Allows you to specify the security policies for each endpoint.

Any changes made here will restart the OPC UA server when you click the
Apply button.

Server URL
Permits you to pick the server endpoint for the security and user token policies explained below. The URL for the service endpoint is constructed from the Protocol, Computer Name/IP, Port and Endpoint Name, shown and/or described above.

Security Policies
- None
  - Authentication, but no encryption.
- Basic128Rsa15
  - Authentication and encryption (AES, key length 128).
- Basic256
  - Authentication and encryption (AES, key length 256).

User Token Policies
The authentication options available, which are used when starting a session. Multiple options can be selected, and are applied consecutively.

- Anonymous
  - No authentication.
- UserName
  - Authenticated with a user name and password.
- Certificate
  - Authenticated using a certificate.

Continue to accept server certificate when it expires
This option allows a UA certificate to be accepted outside of its valid time window, meaning that expired certificates can continue to be used. Checking this box also keeps the UA server and client connected if their system clocks ever get out of synch. And this option also supports connectivity for OPC UA clients running on embedded systems without system clocks, or whose system clocks cannot be adjusted.

If you are using the http protocol along with a security policy, then the clocks on the UA server and client machine must match within 5 minutes at all times. This is a requirement of the WCF subsystem that implements the HTTP security. If you are not able to synchronize the clocks on the server and client machines this closely, you should try either the opc:tcp or https protocol, which do not rely on WCF for the underlying security and so do not exhibit this problem.
Client Certificate Receiving

Automatically accept untrusted certificates

Checking this box will allow any client on the network to connect to the DataHub without verification. Use with extreme caution.

If this option is selected, any client attempting to connect will be accepted temporarily. Its certificate will be placed in the Temporary Certificate Store, and stay there as long as the DataHub continues to run. When the DataHub shuts down, all certificates in the Temporary Certificate Store get deleted. To become permanent, a certificate in the Temporary Certificate Store must be accepted, which puts it into the OPC UA Client Certificate Store. Any client whose certificate is in the OPC UA Client Certificate Store can connect whenever the DataHub is running, whether Automatically accept untrusted certificates is selected or not.

Continue to accept client certificates when they expire

This option allows a UA certificate to be accepted outside of its valid time window, meaning that expired certificates can continue to be used. Checking this box also keeps the UA server and client connected if their system clocks ever get out of synch. And this option also supports connectivity for OPC UA clients running on embedded systems without system clocks, or whose system clocks cannot be adjusted.

If you are using the http protocol along with a security policy, then the clocks on the UA server and client machine must match within 5 minutes at all times. This is a requirement of the WCF subsystem that implements the HTTP security. If you are not able to synchronize the clocks on the server and client machines this closely, you should try either the opc:tcp or https protocol, which do not rely on WCF for the underlying security and so do not exhibit this problem.

Configure Data Domains

Any changes made here will restart the OPC UA server when you click the Apply button.
Provides a way to configure groups associated with particular data domains. These are the same groups configured in the DataHub Security feature. Clicking the Configure button for any group opens the Configure Data Domains window:

Check the boxes of the DataHub domains that you would like to expose to OPC UA clients for this group. The **Server** domain contains diagnostic information about the DataHub's OPC UA server, which you may or may not wish to share with OPC clients. In the **Selected Nodes**, the **NodeId** is a unique identifier that is managed by the OPC UA server.

**Advanced**

Changes made here will **not** restart the OPC UA server.

**Server Diagnostics**

Allows you to enable or disable diagnostics, which may appear in the Event Log or node configuration.

**Operating Limits**

The **Operating Limits** allow you to limit the number of **sessions** and **subscriptions** on the UA server. A session is a connection, made over a secure channel, which of-
fers the UA client means to create one or more subscriptions. Each subscription is a selection of monitored items, or in our case, DataHub points. If either of these boxes is not checked, then no limit is applied.

**Options**

- The **Allow connections when the client and server clocks do not match** option helps to ensure a connection gets made despite any differences in clock times between the client and server.

- The **Automatically create unknown items requested by the client** option is provided to allow the DataHub to dynamically add items as clients request them. To be OPC compliant, the DataHub would normally return an error if a client requests an item that does not currently exist. The primary purpose of this option is to eliminate a start-up race condition where a client using a data item starts before the data item is available from its source. The DataHub will create the item with a null value and bad quality, and return it to the client. Later, when the item's source becomes available, the DataHub will be able to update the client with the correct value.

This box must be **unchecked** for the DataHub to be fully OPC UA compliant.

**Registration of UA Server**

The **Discovery Server URL** default entry of `localhost:4840` is used when this DataHub acts as the Discovery Server. As 4840 is the OPC UA specified port for a Discovery Server, we don’t recommend changing it without good reason.

**HTTP Access Rules**

A list of HTTP endpoints for which permission is granted to OPC UA clients to connect.
These cannot be added to or modified, but can be removed from the list by selecting one or more, and pressing the **Delete** button.

**SSL/TLS Bindings**

![SSL/TLS Bindings](image)

Provides the details about available SSL/TLS bindings, including the **IP Address**, **Port**, and **Subject Name**, in which **CN** is the "Common Name" or application name, and **DC** is the machine name. The **Thumbprint** is the output of the certificate's hash function.

**Server Information**

![Server Information](image)

**Status**
- Indicates the server status, such as **Shutdown**, **Start Server**, **Running**, **Stopping**, etc.

**Sessions**
- The total number of connections from all UA clients, each of which may include one or more subscriptions.

**Subscriptions**
- The total number of subscriptions from all clients, each of which may contain one or more selections of monitored DataHub points. Subscriptions may migrate from one session to another, if a session gets terminated.

**Server Status...**
- This button opens the OPC UA Server Status window:
Properties Window

This window provides more information about each session. Clicking on a session displays details about each of its subscriptions.

Certificates

![Certificate Management Interface]

OPC UA security is managed using certificates. When an OPC UA client and server communicate, they exchange certificates to ensure each other's validity. The following options let you determine which certificates are used, and how they are managed.

Rejected Certificates

The number of security certificates in the Rejected Certificate Store. This status can be changed by clicking Accept All, or Manage Certificates.

Temporary Certificates

The number of security certificates in the Temporary Certificate Store. This status can be changed by clicking Accept All, or Manage Certificates.

Manage Certificates

This button opens the Manage Certificates in Certificate Store window:

![Certificate Management Window]

Certificate Store

Lets you choose which certificate store to display:

- **Rejected Certificates** (not trusted)
- **OPC UA Private Certificates** (not in the Windows certificate store)
• **OPC UA Global Certificates** (in the Windows certificate store)
• **Certificate Authorities**
• **Temporary Certificates** (valid for this session only)

**Filters**
Allow you to filter the certificates listed according to name, domain, and issuer name, as well as certificate type, and whether the certificate has a private key.

**Actions**
Using the buttons or the right-click context menu, you can **Import** other certificates, or select a certificate and **View**, **Delete**, **Reject** it. You can also **Accept** it for the private store or **Global Accept** it for the Windows certificate store.

**Reload Configuration**
Reloads your entire OPC UA configuration

**Application Certificate**
Allows you to view the certificate assigned to the DataHub.

**Store Type**
The type of store for this certificate.

**Store Path**
The file path to the directory store. If the **Store Type** is Directory, it will be in the
Properties Window

path.

**Application Name**
The name of the application, typically Cogent DataHub Data Access Server.

**Organization**
The organization name.

**Application URI**
A URI that uniquely identifies the application.

**Domains**
The host name.

**Subject Name**
The subject name of the certificate, in which `cn` is the “Common Name” or application name, and `dc` is the machine name.

**Issuer Name**
The publisher of the **Subject Name** that issued the certificate.

**Valid Period**
The period of time that the certificate is valid.

**Thumbprint**
The certificate’s thumbprint.

**Details**
Opens the View Certificate Details window that displays the above information, and additional details.

**Export**
Writes the certificate to a file in your file system, without a key. This is necessary when a UA application needs to manually install the certificate.

**Assign**
Lets you assign a different certificate to the DataHub. It must contain a private key to be able to be assigned.

**Regenerate**
Regenerates the certificate. This is useful if you think that the certificate has been compromised, or if you change your computer name. Regenerating the certificate automatically restarts the OPC UA server.

**OPC DA**
The OPC DA option lets you configure the Cogent DataHub to act as an OPC DA server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to the section called “OPC Protocol” and Appendix F, *OPC Overview*.
For step-by-step instructions to configure the DataHub as an OPC DA client, please refer to the section called “Connect to an OPC DA server”.

Click here to watch a video.

**OPC DA Client**

The Cogent DataHub can act as a client to one or more OPC DA servers.

![ OPC Classic Data Access (DA) Configuration](image)

Check the **Act as an OPC Client** box for OPC DA client functionality. Since the Cogent DataHub can be a client to more than one OPC server, you need to specify server information for each OPC client connection. Once you have a server listed, you can activate or deactivate the connection using its **On** check box.

To add a server, press the **Add** button to open the **Define OPC Server** window described below. To edit a server, double-click it or select it and press the **Edit** button to open that window. To remove a server, highlight it and click the **Remove** button.

Pressing the **Reload Data from All Servers** button causes the DataHub to disconnect from all OPC DA servers, and then reconnect and refresh the data set for each server.

**The Define OPC Server Window**

To define or redefine an OPC DA server connection, click the **Add** or **Edit** button to open the **Define OPC Server** Window:

![ Define OPC Server](image)

**Connection Name**

A name used by the DataHub to identify the connection. There should be no spaces
in the name. It doesn’t matter what name is chosen, but it should be unique to other
connection names.

**Computer Name**
The name or IP address of the computer running the OPC server you want to connect
to. Select it from the drop-down list, or type it in.

**OPC Server Name**
The name of the OPC server that you are connecting to, selected from the list of avail-
able servers.

**Data Domain Name**
The name of the DataHub domain in which the data points are received.

**Data Transfer**
The Cogent DataHub supports OPC DA 2.0 and OPC DA 3.0 client protocols. DA 3.0 sup-
port consists of browsing support and support for the **WriteVQT** (Value, Quality, Time-
stamp) methods of the DA 3.0 specification. Normally, the DataHub will determine
whether a particular server is DA 3.0 compliant based on the registry entries made by the
server when it was installed.

If the server is DA 3.0 compliant, then the DataHub will always use DA 3.0 browsing, as it is
substantially faster than DA 2.0 browsing. If the server claims to be DA 3.0 compliant, but
does not offer the DA 3.0 browsing interface, the DataHub will attempt to drop back to the
DA 2.0 browsing interface.

In some cases, the server's DA 3.0 compliance cannot be determined.
This is true if the server name is specified as a GUID in the form
{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnn} where each \( n \) is a hexadecimal digit. In
this case, the DataHub will default to only use the OPC DA 2.0 browsing interface. You can
force the use of DA 3.0 or DA 2.0 using the respective option, as explained below.

For testing purposes, there is also a registry key that can be used to globally
override the use of DA 3.0 browsing for all OPC connections. If the **DWORD** reg-
istry value:

```
HKEY_CURRENT_USER\Software\Cogent\Cogent DataHub\BrowseDA3
```

exists, its value will be interpreted as follows:

- **1**: Always use DA 3.0 browsing, regardless of the server settings
- **0**: Never use DA 3.0 browsing, regardless of the server settings

Since this key is global to all OPC client connections, it should not be created
at all unless a particular testing scenario requires it. The setting will take effect
the next time a connection to the OPC server is made.

With all of these considerations in mind, you have several options for specifying how the
data is to be transferred:
**Maximum update rate (milliseconds)**

This option lets you specify an update rate, useful for slowing down the rate of incoming data. The default is 0, which causes values to be updated as soon as possible. This value is also the polling time used by asynchronous and synchronous reads (see below).

**Read Method**

Choose how to read data from the OPC server:

- **Asynchronous Advise**  The OPC server sends a configured point's data to the DataHub immediately whenever the point changes value. This is the most efficient option, and has the least latency.

- **Asynchronous Read**  The DataHub polls the OPC server for all configured points on a timed interval (set by the **Maximum update rate**). This option is less efficient than Asynchronous Advise, and has higher latency.

- **Synchronous Cache Read**  The DataHub polls the OPC server for all configured points on a timed interval (set by the **Maximum update rate**), and this thread waits for a reply. This option is less efficient than Asynchronous Advise or Read, and has higher latency than either of them.

- **Synchronous Device Read**  The DataHub polls the PLC or other hardware device connected to the OPC server for all configured points on a timed interval (set by the **Maximum update rate**), and this thread waits for a reply. This is the least efficient of all of these options, and has the highest latency.

**Write Method**

Choose how to write data to the OPC server:

- **Asynchronous Write**  The DataHub writes to the OPC server and does not wait for a response. This provides the highest overall performance.

- **Synchronous Write**  The DataHub writes to the OPC server and waits for a response each time. This elicits a quicker response for a given item from the OPC server, but results in lower overall performance. This option is useful if the OPC server doesn't support asynchronous writes at all, or if it can't handle a large number of them.

For these options, the DA 2.0 write methods only transmit a point's value, allowing the server to assign a quality and timestamp as it sees fit. The DA 3.0 methods (**WriteVQT**, supported by DA 3.0 servers only) transmit the Value, Quality, and Timestamp of a point.

**Options**

There are several optional entries:
Treat OPC item properties as DataHub points
This option lets you register and use each OPC item property as a point in the DataHub.

Some OPC servers are slow to register their OPC items and properties. Using this option with one of these servers can significantly slow the start-up time of the DataHub

Read only: Mark all items as Read-Only
Here you can specify that the connection to the OPC server be read-only, regardless of how individual items are specified. Items in the DataHub that originate from such an OPC server will be read-only to all DataHub clients.

Only transmit GOOD quality data to this server
This option prevents any data except that with a quality of Good from being sent to the OPC server.

Replace item time stamps with local clock time
This option allows you to set the timestamps for the items from this server to local clock time.

Force connection to use OPC DA 3.0
This setting will allow the user to choose the DA 3.0 write methods from the Write Method drop-down box. It will also instruct the DataHub to attempt to browse the server using DA 3.0 browsing. This setting will override any automatic information that the DataHub may determine about the server based on the server's registry entries.

Never use OPC DA 3.0
This setting will remove the DA 3.0 write methods from the Write Method drop-down box, and will instruct the DataHub to only use DA 2.0 browsing. This setting will override any automatic information that the Cogent DataHub may determine about the server based on the server's registry entries.

Set failed incoming values to zero
The OPC spec requires an OPC server to send an EMPTY (zero) value whenever it sends a failure code in response to an item change or a read request. Some OPC servers, however, send a valid value with the failure code under certain circum-
stances. To ignore any such value from the OPC server and assume EMPTY, keep this box checked (the default). If instead you want to use the value supplied by your OPC server, uncheck this box.

Unchecking this box will make the DataHub's behavior non-compliant with the OPC specification.

**Never use OPC DA 2.0 BROWSE_TO function**

This setting will disallow the BROWSE_TO function when communicating with OPC DA 2 servers. Sometimes an OPC server will have problems with this function that prevent the DataHub from connecting to it. Checking this box might allow the connection to be established in those cases.

**Never attach to an in-process COM server**

Most vendors include both an in-process and out-of-process COM server with their OPC server installation. If both options are available, the DataHub connects to the in-process server, as it is generally the better choice. This option forces the DataHub to consider only out-of-process servers.

Why is this useful? An in-process server is implemented as a DLL that is loaded into the client's address space. This makes the client very dependent on the good implementation of the server. If there is a crash in an in-process server, the client also crashes. An out-of-process server is implemented as a separate executable. The client communicates with an out-of-process server using the inter-process communication mechanisms in DCOM. In theory an in-process server will be faster than an out-of-process server, but sometimes the in-process server is less robust than the out-of-process server and leads to instability or malfunction in the client.

**Allow VT_EMPTY canonical type for OPC DA2**

The VT_EMPTY canonical type may be incompatible for a particular combination of OPC server and client. For example, some clients or servers that were built before 64-bit integers were common may fail when presented with a 64-bit number. These options (DA2 and DA3) allow you to enable or disable the VT_EMPTY canonical type, either for trouble-shooting or as a permanent part of your configuration.

**Allow VT_EMPTY canonical type for OPC DA3**

See above.

**Wait for server running state**

Every OPC server takes a little time to initialize before it will allow client connections. This option lets the user specify the time to wait for the OPC server to initialize. The wait time is a maximum; if a server initializes before this time, the DataHub will connect right away. If the server doesn't initialize within this time, the DataHub will report this in the Event Log, and then try to connect anyway.

**Pause before reading data**

This parameter specifies a time for the DataHub to pause before reading the OPC server's data set. Some OPC servers report that they are running, but have not yet received the full data set from the process. If the DataHub attempts to connect right
away, it might get a partial data set. The pause is fixed; it will always last for the full time specified.

The two above times are added together. The DataHub will wait until the server is initialized (or until the specified "wait" period is complete) and then pause for the specified "pause" time, before trying to read data from the server. For example, with the defaults of 5000 and 1000, at least 1 second and at most 6 seconds will elapse before the DataHub tries to read the data set.

**Item Selection**

You can select all items, filter for specific items, or select items manually.

![Image of Item Selection](Image)

**Manually Select Items**

Check the **Manually Select Items** box and press the **Configure Items** button to open the OPC Item Selection window, where you can specify exactly which points you wish to use:

![Image of OPC Item Selection](Image)

You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. Follow these guidelines for making selections:

- To select a server item from the right-hand pane, click its check-box.
- To highlight a list of consecutive server items, click the first item, hold down the **Shift** key, and then click the last item. To highlight separate server items, hold down the **Ctrl** key as you click each item. To select a group of highlighted items, use the **Spacebar**.
- Selecting a server item does not automatically add any of its child items. Each child item must be added separately. To view child items, click the + sign in front of the item. If an item has one or more children that have been selected, the item name(s) will appear in bold.
• To delete selected items from the right-hand pane, highlight them and press the **Remove Selected** button. Use the **Shift** and **Ctrl** keys as above to highlight groups of selected items.

![Dynamic Items Window](image)

You may also configure dynamic items on the server. As you type in the **Server Item ID**, the system will fill in an identical **DataHub Point Name** for you (which you can change at any time). Press the **Enter** key or the **Apply** button to create the item. Checking the **Copy names from selection** box will fill in the entry with the name you select from the **Selected Items** list (above). The **Recognize branch delimiter in point name** option lets you select and apply a point delimiter for your dynamic items.

**Load All Items on Server**

In addition to manually loading items, you have the option in the Define OPC Server dialog to register all points, or filter for groups of points, from the OPC server.

![Server specific item filters](image)

In the **Server specific item filters** area you can enter one or more strings to filter for groups of items in the OPC server. Use the **Add** or **Edit** button to open the **Edit a filter string** window:

![Edit a filter string](image)

Enter a string that matches an item name, or a pattern to match multiple names. Each OPC server has its own syntax for pattern matching, so you may have to experiment a little to get exactly the points you need. Commonly, the symbol * matches any number of characters, while the symbol ? often matches a single character. In that case, an entry of ?a* would bring in all items with a as the second letter in their names.

**OPC DA Server**

The Cogent DataHub can act as a server to any number of OPC DA clients.
Check the **Act as an OPC Server** box to have the Cogent DataHub function as an OPC DA server.

If your OPC client requires that you hand-enter the OPC server name, use either `Cogent.CogentDataHub` or `Cogent.CogentDataHub.1`.

The **Do not adjust OPC registry entries for this program** option tells the DataHub not to alter its registry settings. This is useful if you want to use the DataHub with a redundancy server or some other program that modifies the DataHub’s registry independently. Without this box checked, the DataHub will overwrite any external changes when it starts or when a change to the **Act as an OPC Server** status is applied.

These two boxes work together, because turning the OPC server behavior on or off necessarily makes changes to the registry. Here is how you can change OPC DA server behavior when you also need to maintain registry settings:

1. Uncheck **Do not adjust OPC registry entries for this program**. This will make the **Act as an OPC Server** checkbox visible.
2. Check or uncheck the **Act as an OPC Server** as needed, and click **Apply**.
3. Check **Do not adjust OPC registry entries for this program** and click **Apply**.

The **Allow VT_EMPTY canonical type for OPC DA2 / DA3** options allow the DataHub to send VT_EMPTY canonical data types for OPC DA2 clients, OPC DA3 clients, or both. By default the DataHub does not send data with a canonical type of VT_EMPTY because many OPC DA2 clients will not accept that data type.

Leaving the **Allow 64-bit types when client requests VT_EMPTY types** option unchecked forces the server to send all values of VT_EMPTY canonical type as 32-bit numbers.

Normally clients tell the server what number format they intend to accept, and the server has the responsibility to provide that number format. However, a client can send VT EMPTY as the requested type, which means that the client will accept any number format. Some clients that do this were built before 64-bit integers became common, and fail when presented with a 64-bit number, even when they have requested VT EMPTY. Keeping this box unchecked (the default setting) prevents this from happening. The **Allow VT_EMPTY canonical type for OPC DA2 / DA3** options allow the DataHub to send VT EMPTY canonical data types for OPC DA2 clients, OPC DA3 clients, or both. By default the DataHub does not send data with a canonical type of VT EMPTY because many OPC DA2 clients will not accept that data type.
The **Automatically create unknown items requested by the client** option is provided to allow the DataHub to dynamically add items as clients request them. To be OPC compliant, the DataHub would normally return an error if a client requests an item that does not currently exist. The primary purpose of this option is to eliminate a start-up race condition where a client using a data item starts before the data item is available from its source. The DataHub will create the item with a null value and bad quality, and return it to the client. Later, when the item’s source becomes available, the DataHub will be able to update the client with the correct value. If the OPC client requests an item ID without a colon character (:) to denote a domain, the DataHub will assign the domain `dynamic`, inserting a point called `dynamic:pointname`, and return success.

**COM Security**

If you need to connect the Cogent DataHub over a network and for some reason you can’t use tunnelling, here is an option to facilitate COM configuration.

Check this box to relax COM security. This setting will override the COM permission settings for the application, but will not override the system’s global COM restrictions. It is common for OPC DA servers to operate at minimal DCOM security settings, since high security interferes with connectivity and most control systems do not operate in hostile network environments. If in doubt, consult your system administrator.

**OPC A&E**

The **OPC A&E** option lets you configure the Cogent DataHub to act as an OPC A&E server, an OPC A&E client, or both simultaneously.

**OPC A&E Client**

The Cogent DataHub can act as a client to one or more OPC A&E servers.
Check the **Act as an OPC A&E Client** box for OPC A&E client functionality. Since the DataHub can be a client to more than one OPC A&E server, you need to specify server information for each OPC A&E client connection. Once you have a server listed, you can activate or deactivate the connection using its **On** check box.

To add a server, press the **Add** button to open the **Configure OPC A&E Server** window described below. To edit a server, double-click it or select it and press the **Edit** button to open that window. To remove a server, highlight it and click the **Remove** button.

Pressing the **Reload Data from All Servers** button causes the DataHub to disconnect from the A&E servers and then re-establish the connection and re-query the alarm and event information from each of them, just like a new connection.

### The Configure A&E Server Window

To define or redefine an OPC A&E server connection, click the **Add** or **Edit** button to open the **Define OPC A&E Server** Window:

![Configure OPC A&E Server Window](image)

#### Connection Name

A name used by the Cogent DataHub to identify the connection. There should be no spaces in the name. It doesn't matter what name is chosen, but it should be unique to other connection names.

#### Computer Name

The name or IP address of the computer running the OPC A&E server you want to connect to. Select it from the drop-down list, or type it in.

#### OPC Server Name

The name of the OPC A&E server that you are connecting to, selected from the list of available servers.

#### Data Domain Name

The name of the DataHub domain in which the data points are received.

#### Connection Delay

The number of milliseconds to delay the initial connection.

#### Retry Delay

The number of milliseconds to wait before retrying a failed connection.
Options

You have several additional options:

**Mark all items as Read-Only and disable acknowledgements to this server**
This option will protect items on the Cogent DataHub from being changed by the client.

**Accept incomplete acknowledgement information**
This allows connections to OPC A&E clients which are not configured for acknowledgements or that don't support this part of the OPC A&E specification.

**Accept out of order condition events**
In OPC A&E the time stamp of an event should always be newer than the time stamp of the previous event for any condition. If, perhaps due to differing clock times on connected DataHub machines, you get errors indicating that events have arrived with out-of-order time stamps, then you can select this option to eliminate the warnings and accept the events.

**Do not load initial condition state from the server**
When the DataHub connects to an A&E server it queries the server for the current state of all alarm conditions. Some servers report all conditions, whether they are active or not. Other servers only report the state of active conditions. Other servers do not report their initial state at all, and simply let the client learn about condition states as they change.

Selecting this option tells the DataHub to wait for an event before identifying conditions (and sources). That means that it won't call GetQualifiedSourceName, and therefore will always present the source as seen in szSource. It also means that an A&E client will not be able to browse the complete condition tree from the DataHub, as the DataHub will start with no condition information and will only discover it as events arrive.

**Do not add area name to source names**
When constructing a source name from the initial condition state, the DataHub builds the source name by appending the source's area path followed by the unqualified source name from the server. This option disables this behaviour and uses only the unqualified name as the source name.

If you are using an A&E server that provides only a partial list of conditions when reading the initial condition state then this allows you to adjust for the behaviour of the server when discovering new conditions as events occur. Specifically, some servers...
provide an unqualified source name when requesting the initial condition state, but a qualified source name in subsequent events. This option allows you to produce source names that are consistent regardless of whether the source was determined during initialization or discovered later.

**Make A&E Status available as individual data points**

This option gives you a way to maintain and access the values of A&E status variables in Cogent DataHub data points.

**Filters**

There are several options for filtering alarms and events:

![Filters screenshot](image)

**Filter by event type**

There are three options available:

- **Simple** events are not related to an alarm, and cannot be tracked.
- **Tracking** events originate outside the process being monitored, for example, an operator intervention.
- **Condition** events indicate that an alarm has been triggered. These events can be activated or deactivated, and they have an acknowledgement mechanism.

**Filter by severity**

*Severity*, or priority, indicates the urgency of a condition for an alarm. A low value, such as 1, corresponds to a low urgency event, for example an informational message. A high value, such as 1000 represents an extreme emergency condition.

**Filter by category:**

This filter lets you select alarms or events according to the type of event, or event category.

**Filter by area:**

This filter lets you select alarms or events based on the *area*, which is typically a location in a plant, or a specific machine.

**Filter by source:**

This filter lets you select alarms or events based on the specific OPC A&E tag for the point.

**OPC A&E Server**
The Cogent DataHub can act as an OPC A&E server to any number of OPC A&E clients. It cannot be configured to act as a source for generating alarms and events, but if the Cogent DataHub is configured as an A&E client (above), this A&E server configuration allows it to pass along A&E values. This is useful for:

- Tunnelling OPC A&E data.
- Converting OPC A&E data to OPC DA data.
- Allowing OPC DA clients to interact with and display data from OPC A&E servers.

Check the **Act as an OPC A&E Server** box to have the Cogent DataHub function as an OPC A&E server, and choose a data domain. There are two optional settings:

**Accept incomplete acknowledgement information**
Allows for the receipt of incomplete acknowledgements.

**Send a shutdown to clients when event configuration changes.**
This option may allow a client to reload events after they have been modified, and/or add new events.

**Tunnel/Mirror**

The Tunnel/Mirror option lets you configure the Cogent DataHub to act as a master or slave for tunnelling/mirroring. Tunnelling/Mirroring allows you to send any data in the DataHub across a network robustly and securely. Tunnelling is done using DHTP over TCP, which provides connectivity across a network or over the Internet.

**Tunnelling and Mirroring**

The Cogent DataHub tunnelling connection is sometimes referred to as a **mirroring** connection. Mirroring means that the data and any updates to that data on one DataHub are exactly mirrored across the network onto the other DataHub, and vice-versa. For all practical purposes, tunnelling and mirroring are identical. People working with OPC tend to use the term “tunnelling” while people from other backgrounds often say “mirroring”. So Cogent uses “tunnelling” for the Cogent DataHub, and “mirroring” for other Cogent products.
**Direct TCP connections**

In addition to tunnelling, the Cogent DataHub can accept direct DHTP connections from any TCP client using the DataHub APIs for C++, Java, and .NET, such as DataSim or other, custom applications.

**Master and Slave**

We identify the two tunnelling/mirroring DataHubs as *master* and *slave*. The only difference between the master DataHub and slave DataHub is that the slave initiates the connection. Once the connection is established, they function exactly the same. It is possible for a DataHub to be both tunnelling/mirroring slave and master simultaneously—acting as a slave to one or more DataHubs and a master to one or more others. For slave mode you need to specify each master.

**Tunnel/Mirror Slave**

Check the **Act as a tunnelling/mirror slave to these masters** box to have the Cogent DataHub act as a slave.

To add a master for this mode, click the **Add Master...** button. To edit a master, double-click it, or select it and press the **Edit...** button. Either button opens the **Tunnel/Mirror Master** window:
Type in the following information:

**Primary/Secondary Host**

The name or IP address of the host computer. This slave DataHub will alternate attempts to connect first on the primary host, then on the secondary host, back and forth until a connection is made. The secondary host is optional, and if not entered, all attempts to reconnect will be on the primary host. If the connection is interrupted, the DataHub will again alternate attempts at reconnection on the primary and secondary hosts.

This feature is not recommended for redundancy because it only checks for a TCP disconnect. The DataHub **Redundancy** feature, on the other hand, provides full-time TCP connections to both data sources, for instantaneous switchover when one source fails for any reason. There is no need to start up the OPC server and wait for it to configure its data set. You can also specify a preferred source, and automatically switch back to that data source whenever it becomes available. By contrast, the primary and secondary host in the tunnel can act as a primitive form of redundancy, but will only switch on a connection failure at the TCP level, which is only one sort of failure that a real redundancy pair must consider.

**Port**

The port number or service name as entered in the **Master service/port** entry box of the master on the remote computer.

**Local data domain**

The local Cogent DataHub data domain for this slave. It is common, but not necessary, to create or use an existing local data domain that has the same name as the remote data domain.

**Remote data domain**

The name of the remote Cascade DataHub data domain, which is the tunnelling master. Point names will be mapped from that data domain into the local data domain, and vice versa.

**Remote user name**

The user name for TCP security, established on the tunnelling master, using the DataHub **Security** option in the Properties window.

**Remote password**

The password for TCP security, established on the tunnelling master, using the DataHub **Security** option in the Properties window.

**Secure (SSL)**

You can establish a secure connection using SSL tunnelling as long as the tunnelling master DataHub you are attempting to connect to has been configured for secure connections. (See below.) Selecting **Reject invalid certificate** causes the DataHub to check that the certificate date is valid, and the certificate chain is trusted. Selecting **Reject host name mismatch** will have the DataHub check that the certificate subject matches the host
name. If you select neither of these two boxes then any certificate will be accepted. This is not recommended because it is not good security, but it can be useful to test the SSL connection. For more about SSL, please refer to the section called “SSL and Firewalls”.

**WebSocket**

You can connect via WebSocket. This option is applied for both primary and secondary hosts, and allows you to enter a **Proxy address**, and a **Proxy port** number, a **username**, and a **password** as needed. When tunnelling through a proxy, HTTP uses normal HTTP proxy, and HTTPS uses HTTP CONNECT proxy. You can select the **Always use HTTP CONNECT** to use it for HTTP as well as HTTPS.

The WebSocket protocol requires a web server to act as an intermediary. So, for this option you will need to use the DataHub Web Server on the tunnelling master DataHub (as explained below).

There are several options for the mirrored connection.

1. **Data Flow Direction** lets you determine which way the data flows. The default is read-only data flow from master to slave, but you can set up a read-write or write-only connection by choosing those options.

   To optimize throughput, check the **Read-only: Receive data from the Master, but do not send** option. Only do this if you actually want a read-only connection. If you do not require read-write access, a read-only tunnel will be faster.

2. **When the connection is initiated** determines how the values from the points are assigned when the slave first connects to the master. There three possibilities: the
slave gets all values from the master (the default), the slave sends all its values to the master, or the data from master and slave gets synchronized. The availability of these options depends on the data flow direction selected above.

3. **When the connection is lost** determines where to display the data quality as "Not Connected", on the master, on the slave, or neither.

   If you have configured **When the connection is initiated** as **Synchronize based on time stamp** (see above), then this option must be set to **Do not modify the data quality here or on the Master** to get correct data synchronization.

4. **Connection Properties** gives you these options:
   - **Replace incoming timestamp...** lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.
   - **Transmit point changes in binary** gives users of x86 CPUs a way to speed up the data transfer rate. Selecting this option can improve maximum throughput by up to 50%, depending on the type of data being transmitted. This option uses a more efficient message encoding scheme than the default ASCII encoding, but it will only work if both sides of the tunnel are running on an x86 architecture CPU. This would be typical of Windows communicating with Linux or QNX, or with another Windows computer. Numeric data benefits most from this option.
   - **Target is an Embedded Toolkit server** allows this slave to connect to an Embedded Toolkit server rather than to another DataHub.
   - **Heartbeat** sends a heartbeat message to the master every number of milliseconds specified here, to verify that the connection is up. Setting this value to 0 stops the heartbeat from being transmitted.
   - **Timeout** specifies the timeout period for the heartbeat. If the slave DataHub doesn't receive a response from the master within this timeout, it drops the connection. You must set the timeout time to at least twice the heartbeat time. Setting this value to 0 will cause the DataHub to rely on the TCP implementation for detecting a broken connection. This can be useful when your network connection is very slow. Please refer to the section called “Tunnel/Mirror (TCP) Heartbeat and Timeout” for details.
   - **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

**Testing the Client Connection**

There is a DataHub running on a Skkynet cloud server that you can connect to for testing. Here are the parameters you will need to enter for it:

- **Primary Host** demo.skkyne.com
- **Port** Will be set automatically by the system, 80 for WebSocket and 443 for Secure (SSL).
• Local data domain  cloud
• Remote data domain  DataPid
• Remote user name  demo/guest
• Remote password  guest
• WebSocket  Must be selected.
• Secure (SSL)  Optional.

License verification timeout

The DataHub gives a TCP connection 30 seconds to verify a license. If your network is too slow, this might cause a licensing error. You can increase the timeout to up to 60 seconds by editing this registry entry:

HKEY_LOCAL_MACHINE\SOFTWARE\Cogent\Cogent DataHub\TCPLicenseTimeoutSecs

Tunnel/Mirror Master

You can configure your DataHub to act as a master for plain-text or secure (SSL) tunnelling, or both. Additionally, the DataHub offers Websocket support for tunnelling, as explained below.

Accept plain text connections on service/port

This option enables plain text connections.

For this and the next option, if you enter a name for the service/port instead of a number, that name must be listed in the Windows services file. Please refer to The Windows Services file Appendix for details.

Accept secure connections on service/port

The DataHub installs an SSL Certificate for you. If you wish to move it or use a different one, you can change the directory path here. The SSL implementation uses the default SSL-3 encryption cipher: DHE-RSA-AES256-SHA. This is a 256-bit encryption. The server and client negotiate the best encryption based on what both can support. The DataHub does not validate the SSL certificate with any outside certificate authority. It uses the SSL connection for encryption only, not authentication.

Try to send data even if it is known to be superceded

You can also configure the master to attempt to send "old" data (superseded by more...
recent data). Check any or all of **Boolean**, **Integer**, **Float**, or **String** that apply to the kind of superseded data that you wish to have sent.

To optimize throughput using this option, please refer to the section called “How to Optimize”.

**WebSocket Connections**

The WebSocket protocol supports real-time bi-directional data transfer over TCP, by maintaining a constant connection over which messages can be passed, in plain text or SSL. The DataHub implementation of WebSocket enables secure connections through network proxies without opening any firewall ports.

The WebSocket protocol requires a web server to act as an intermediary for the connection. So, to support incoming WebSocket connections from DataHub tunnelling clients, you will need to configure the tunnelling master DataHub's **Web Server**. For WebSocket connections, we recommend using SSL, on port 443. A plain text connection is also possible, but is less secure and less likely to pass through a proxy.

**Bridging**

The Bridging option lets you configure the Cogent DataHub to configure data bridging. Bridging means connecting points from two different DataHub clients so that when one point changes, its value gets written to the bridged point.

For general and how-to information about bridging, please refer to **Bridging**.

Click here to watch a video.

You can configure the bridge to be one-way in either direction, or bidirectional. In addition, you can transform the data as it passes through the DataHub, using linear transformations.

**Bridge Activation**

This setting enables and disables bridging globally for the whole Cogent DataHub program.
Ensure the box is checked to enable bridging. Uncheck the box to disable bridging.

Status

This shows the total number of bridges that have been configured, and how many of them are currently active.

![Status Table]

Configure Bridges

Click the **Configure Bridges** button to open the **Bridge Configuration** window:

![Bridge Configuration Window]

The following sections give an overview of bridge configuration. For step-by-step instructions, please refer to the section called “Configuring Bridges”.

Point Selection
This is where you select the points to be bridged—a source point and a destination point.

You can click on the point you need, or enter the name in the data-entry box at the top of the column.

**Storing Transformations**

You can store transformations and retrieve them by name later on.

To save a transformation, click the **Store...** button and enter a name in the pop-up box. Once stored, the transformation will become available by name in the drop-down list.

To load a transformation, simply select its name from the list.

**Transform**

Specify the type of transformation by clicking **Direct copy, Linear Transformation**, or **Linear Range Mapping**.
• **Direct copy** makes no transformations. It just copies the point.

• **Linear Transformation** lets you multiply by one value and add another value, such as in the equation \( y = mx + b \) where the destination point is \( y \), the source point is \( x \), the **multiply by** value is \( m \), and the **then add** value is \( b \). For example to transform a Celsius source point to a Fahrenheit destination point, you would multiply by 1.8 and add 32, or

\[
\text{Fahrenheit} = (1.8 \times \text{Celsius}) + 32
\]

If you have selected the **Inverse** direction for a transformation, you will get the inverse of the transformation. In this example, you would get a conversion from Fahrenheit to Celsius, or the results of this equation:

\[
\text{Celsius} = (\text{Fahrenheit} - 32) / 1.8
\]

As an alternative to entering transformation values, the DataHub also offers **Linear Range Mapping**.

• **Linear Range Mapping** lets you enter a range for the source and destination, and the DataHub automatically calculates the corresponding linear transformation. For example, to create the same Fahrenheit to Celsius transformation, you could use the defaults of 0 and 100 for the **Min** and **Max** of the source point. Then you would enter 32 and 212 for the **Min** and **Max** of the destination point. As soon as you make these entries, the correct values get entered automatically in the **Linear Transformation**.

When you use linear range mapping, you can limit the transformed value to the maximum and minimum by checking the **Clamp** boxes. The clamps get applied to the point being changed, i.e. to the destination point for forward direction, to the source point for inverse direction, and to both points for bidirectional bridges.

**Direction**

Decide which direction you want the bridge to apply.
• Select **Forward** to change the destination point when the source point changes, but not change the source when the destination changes. If you select **Force consistency** with this option, and if the destination point gets changed for some reason, then the DataHub will attempt to force its value to be consistent with the source point value.

• Select **Inverse** to change the source point if the destination point changes, but not vice-versa.

  Selecting **Inverse** will apply the inverse of the transformation, as explained above.

• Select both **Forward** and **Inverse** for a bidirectional bridge, where either point changes whenever the other point changes. This combination will deselect **Force consistency** to eliminate the possibility of conflicting behavior.

Click the **Apply** button to create and activate the bridge. The DataHub will create the bridge and update the bridged points immediately.

### Point Display

Here you can see all the bridges that exist in the system, and the significant information about them.

If you click on a transformation, the source point, destination point, and transform information get displayed in their respective panels. Use the check box at the front of each bridge to activate or deactivate that particular bridge.

### Redundancy

The Redundancy option lets you configure redundant connections to the Cogent DataHub. Please see *Using Redundancy* for more information on how this feature is used.

The purpose of redundancy is to collect data from two data sources and present to the client program a single output data set. The Cogent DataHub will determine which source
will be presented to the client program, and switch between the two sources without affecting the client. The client will only read data from the output data set.

The two input and one output data sets are maintained as separate data domains in the Cogent DataHub. The sources do not need to be the same protocol, so redundancy can be applied to two sources, for example where one is a direct OPC connection and the other is a tunnel.

![Domain Bridging and Redundancy](image)

Check the **Enable redundancy** box to activate this feature. Redundant connections are created and stored in sets. You can create multiple redundant sets, and activate or deactivate each set using its corresponding **On** check box in the list. To edit a redundancy set, double-click it here, or select it and press the **Edit** button to open the **Configure Redundancy** window (see below). To remove a set, highlight it and click the **Remove** button.

To create a redundancy set, press the **Add** button, which opens the **Configure Redundancy** window:

![Configure Redundancy](image)

**Data Domains**

**Label:**
A name used by the Cogent DataHub to identify the redundancy set. There should be no spaces in the label. It doesn't matter what label is chosen, but it should be unique to other labels.

**Source Domain 1:**
The DataHub data domain for the first data source. If this is the preferred source, check the **Preferred source** button.
If a preferred data source has been specified then the DataHub will use that source whenever possible, even if the other source is also available. This is useful if the two data sources have different characteristics. For example, the preferred source may offer a higher bandwidth than the other source. If neither data source is selected as preferred, the DataHub will maintain whichever data source is currently being used until it meets any invalid criteria (see below).

**Source Domain 2:**
The DataHub data domain for the second data source. If this is the preferred source, check the **Preferred source** button.

**Output Domain:**
A name for a DataHub data domain which will be the output of the redundant connection, to which the client will connect. If the output domain does not exist, the DataHub will create it.

**Input Domain is Invalid When**
Entries in this section determine when the DataHub should switch from one redundant data source to the other.

**Data quality is:**
Gives you the option of switching data sources based on a change in data quality for the point(s) you have selected (below). You can set the criteria of equal to or not equal to a list of available qualities, such as:

<table>
<thead>
<tr>
<th>Quality</th>
<th>Data Hub Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad</td>
<td>Equal to Bad</td>
</tr>
<tr>
<td>Comm Failure</td>
<td>Good</td>
</tr>
<tr>
<td>Config Error</td>
<td>Initializing</td>
</tr>
<tr>
<td>Device Failure</td>
<td>Last Known</td>
</tr>
<tr>
<td>EGU Exceeded</td>
<td>Local Override</td>
</tr>
<tr>
<td>Last Usable</td>
<td>Not Connected</td>
</tr>
<tr>
<td>Sensor Calibration</td>
<td>Out of Service</td>
</tr>
<tr>
<td>Sensor Failure</td>
<td>Sub Normal</td>
</tr>
<tr>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>Local Override</td>
<td></td>
</tr>
<tr>
<td>Initializing</td>
<td></td>
</tr>
<tr>
<td>Not Connected</td>
<td></td>
</tr>
<tr>
<td>Out of Service</td>
<td></td>
</tr>
<tr>
<td>Sub Normal</td>
<td></td>
</tr>
<tr>
<td>Uncertain</td>
<td></td>
</tr>
</tbody>
</table>

Generally speaking, all of the above qualities are considered not good except for **Good** and **Local Override**. To ensure that you are getting good quality data from your OPC server, you can switch when **Data quality is not equal to Good**.

**Data value is:**
Gives you the option of switching data sources based on a change in the value of the
data point(s) you have selected (below).

**For point(s)**
Allows you to select which points you want to monitor for quality or value (see above).

**For any point in the domain**
 Lets you monitor all points in the domain and switch when any one of them meets the criteria.

**For this point**
 Lets you specify an individual point name. The point name can be applied to a single point, or to a group of points whose names match the pattern.

For more information please see Configure the switch in the Using Redundancy chapter.

**Data Flow Detection**
For data that changes regularly, **Data Flow Detection** lets you switch data sources whenever a gap in the data flow is detected. This option will watch for any change in any data point from an input domain. If no point in the input domain changes within the specified time, the entire input domain is assumed to be invalid. A subsequent data change must pass the validity checks above for the input to be considered valid.

**Switch sources if data stops for:**
The number of seconds that the data flow from the domain must stop before the DataHub will switch to the redundant data domain.

**Options**

**Never switch. Always use the first source.**
If you never switch, you never use a second source domain. Thus, this allows you to effectively create a copy of a domain, by copying **Source Domain 1** into the **Output Domain**. To have the output domain function as a read-only copy of the source domain, select the **Ignore values written to the output domain** option, as explained
below.

**Do not refresh the output domain when switching.**

Keeps the existing data in the output domain during a switch, and only updates with values from the new domain when a change occurs. Choosing this option may cause a mismatch between input and output domains for an indeterminate length of time.

Normally during a switchover the DataHub copies all values from the new source domain into the output domain. This copying activity might cause delays in updating the output domain for extremely large numbers of points. If that is your situation, and you know that your two servers are synchronized in all meaningful ways, you may wish to select this option. However, you need to keep in mind that values in the input and output domains may not match for an undetermined period of time.

**Never copy the data model to the output domain.**

Preserves the data model of the output domain, or if there is no data model, flattens the data model from the input domain. In either case, the data point names are maintained. This can be helpful if targeting a system with limited system resources, such as an embedded system, or if you have an existing data model on the output domain and do not want it overridden by the data model on the input domains.

**Always copy the data model when switching.**

Normally the output domain tracks changes in the data model, and when a switch occurs, if the data model has changed, it gets rewritten. This option forces the output domain to copy the data model, whether it has changed or not.

**Ignore values written to the output domain.**

Normally, data written to the output domain propagates back to the input domains. This option prevents that from happening. Data written to the output domain will not be written to the input domain. Used with **Never switch. Always use the first source.** (above), this will make the output domain function as a read-only copy of the input domain.

**Do not write backward from output to read-only inputs.**

This option is similar to **Ignore values written to the output domain.** (above), but prevents writes to points in the input domain only for those points marked as read-only. Data written to writable points in the output domain will be written to the input domain.

**Create data points showing input statistics**

Used for debugging, this option creates extra points in the root of the output domain that indicate how many points are considered valid, invalid and uninitialized for each input domain.

The radio buttons under **Treat uninitialized points as:** let you choose whether a point in an input domain that has never been assigned a value (**BAD** quality, 0 timestamp, and 0 value) should be treated as:

- **Normal**: The validity rules apply normally to it.
- **Invalid**: The domain will never be used as an input until all data points have a value assigned to them.
Properties Window

- **Valid**: Any uninitialized points are ignored when determining whether the input domain is valid.

**Status and Control Data Points (blank for disabled)**

- **Point for current source number**: The name of a DataHub point that will indicate which source is in use.
- **Point for current state of domain 1**: The name of a DataHub point that will indicate the state of Domain 1.
- **Point for current state of domain 2**: The name of a DataHub point that will indicate the state of Domain 2.
- **Point for preferred source number**: The name of a DataHub point that will indicate which data source is your preferred source.

Click the **OK** button to submit your entries.

For more information please see Troubleshooting in the Using Redundancy chapter.

**Database**

The Database option lets you configure the DataHub for writing data or making queries to any ODBC-compliant database.

For more information about data logging, please refer to *Write to a Database*, and to learn more about making database queries, please refer to *Query a Database*.

Click here to watch a video.
Write to a Database (ODBC)

Enable logging to ODBC database
Globally enables or disables all ODBC logging activity.

Reconnection delay (s):
The number of seconds before a reconnect is attempted if the ODBC connection is broken.

Maximum transaction queue
The DataHub maintains an in-memory queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. You can modify the depth of this queue to reduce the chance of involving the disk during busy periods. The queue depth for logging defaults to 100 messages.

Show diagnostics in the Script Log
Puts diagnostic messages about the connection in the Script Log.

Configure button
The Configure button opens the ODBC Data Logging Window.

For detailed instructions on using this interface, please refer to Write to a Database.

Store and Forward

The term store and forward refers to a type of database connection where the data is stored locally to disk and then later forwarded to the database. The Cogent DataHub performs an advanced form of store and forward that only writes to disk if the database is not connected, or has been paused. If the database is available, the data will be transmitted directly to the database. This means that there is no penalty for using store and for-
ward during normal operation. The DataHub store and forward mechanism uses two levels of disk caching to ensure that all data gets logged, and nothing is lost.

When the database first becomes available after an outage, the DataHub starts writing cached values to the database, and continues writing new values to the cache. In this way, the values are inserted into the database in the order in which they are generated by the system. Once the cache is cleared, the DataHub then starts writing new values directly to the database.

Enable store and forward
Activates the store and forwarding feature.

Always write queue to disk
Data in the transaction queue will be written to disk cache first, and from there to the database. The safest protection against a crash is to check this box, and uncheck Delay writes to disk (below).

Never write queue to disk
The data in the transaction queue will be only stored in memory, and never written to disk.

Delay writes to disk
Data in the transaction queue will be written to disk at the most opportune times. The safest protection against a crash is to uncheck this box, and check Always write queue to disk (above).

Allow duplicates while forwarding stored data
If the network breaks while transmitting data from a cache, the Cogent DataHub needs to know how to handle any already-sent data when it reconnects. Leaving this box unchecked will require the Cogent DataHub to track its cache position at all times, and modify that information each time a value is sent. This will impact the speed of every transmission, but it will ensure that no values get transmitted twice. Checking this box will cause the DataHub to simply start from the beginning of the queue or cache on each reconnect, and retransmit some data. This significantly reduces data-handling complexity and decreases transmission rates. This option is particularly useful if network breaks are frequent and some duplication of logged data is acceptable.
Show statistics in tray menu

Adds a Data Logging entry to the DataHub's system tray menu, which lets you open a statistics window:

Transactions sent successfully:
The number of transactions that were sent, either directly to the database, or to the disk cache.

Transactions / sec (10 sec window):
The sending rate for transactions, calculated over the past 10 seconds.

Transactions currently on queue:
The number of transactions in the queue.

Results currently on queue:
Not yet documented.

Transactions rejected (full queue)
The number of transactions that were rejected from the queue because it was full.

Transactions stored in L1 cache
The number of transactions taken off the queue and put into the first-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L1 cache
The number of transactions that were not able to be stored in the first-level cache.

Transactions stored in L2 cache
The number of transactions taken off the queue and put into the second-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L2 cache
The number of transactions that were not able to be stored in the second-level cache.
Transactions forwarded from cache
The total number of transactions forwarded from both caches. This number should be the sum of L1 and L2, once all transactions have been forwarded, and as long as the DataHub was started up with no cache on disk.

Transactions failed from cache
The number of transactions attempted from cache, could not be successfully delivered, and were stored for later transmission. This phenomenon may occur the first time that the DataHub learns that the database is not available. For example, you'll see this for every network break if you've checked **Always write queue to disk**.

Cache directory:
The path and directory name for the cache.

Maximum cache size (MB):
The amount of disk space to allocate for the cache, in megabytes.

Query a Database (ODBC)
In addition to writing data to a database, the Cogent DataHub also allows you to query a database and put the resulting values into the DataHub.

Enable query from ODBC database
Globally enables or disables all ODBC query activity.

Reconnection delay (s):
Specifies the number of seconds before a reconnect is attempted if the ODBC connection is broken.

Maximum transaction queue
The DataHub maintains an in-memory queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. You can modify the depth of this queue to reduce the chance of involving the disk during busy periods. The queue depth for queries defaults to 2 messages.

Show diagnostics in the Script Log
Puts diagnostic messages about the connection in the **Script Log**.

Show statistics controls in tray menu.
Creates a new entry in the DataHub system tray menu that lets you open a statistics control window.
Configure button

The **Configure** button opens the ODBC Data Logging Window.

For detailed instructions on using this interface, please refer to *Query a Database*.

Web Server

The Web Server option lets you configure the DataHub to run as a lightweight http server capable of serving HTML documents, Java applets, and many kinds of binary files. It features password-protected access and server-side scripting, and supports DataHub WebView.

For information about using the Web Server, please refer to *Using the Web Server*.
Web Server

The web server can be turned on and off while the DataHub is running.

Ensure the Act as web server box is checked to enable the web server. Uncheck the box to disable it.

Base Configuration

Port

The number of the port used to make TCP connections to the web server. The DataHub web server is preconfigured to run on port number 80, but you might need to change that setting.

Windows allows multiple users on a single TCP port, and never refuses a connection. However, this can cause irregular behavior. It is essential that the DataHub web server be the exclusive user of a port. Please refer to step 4 in the section called “Configuring the DataHub Web Server” for more details.

Use secure sockets (SSL)

Checking this box will cause the web server to run in secure mode.

Allow directory listing

Checking this box allows users to browse the directory on your server.

Document root directory

The root directory for your documents.

Options

Error log file: C:\Users\Bob\AppData\Roaming\Cognet DataHub\log\error.log
Access log file: C:\Users\Bob\AppData\Roaming\Cognet DataHub\log\access.log
SSL certificate file: C:\Program Files (x86)\Cognet DataHub\plugin\WebServer\etc\id...
Error log file:
The path and name of the file where errors are logged.

Access log file:
The path and name of the file where access attempts, successes, and failures are logged.

SSL certificate file:
The path and name of the certificate file used for secure sockets (SSL). Please see SSL Certificates for more information about SSL certificates in the DataHub.

Require a password for all requests
Applies password security as configured in Security.

The security model changed from version 7.1 to version 7.2. User names and passwords in the Security tab will be maintained when moving from V7.1 to V7.2. User names and passwords in the Web Server tab will be lost. When upgrading to V7.2 you must re-assign Web Server realms to any relevant users in the Security tab by clicking on the password field for that user. When reverting from V7.2 to V7.1, you must re-enter the path to the Web Server authentication file that you had earlier used with V7.1.

Authentication realm
The name of the current authorization realm used for password verification.

DataHub WebView

DataHub WebView is a data visualization tool that is available for two platforms:

- Desktop WebView runs on Windows Desktop.
- Silverlight WebView runs on Internet Explorer.

This section covers the necessary configuration parameters for Silverlight DataHub WebView only. For information about Desktop WebView and how to use DataHub WebView, please refer to the DataHub WebView manual.
Data Domains Visible to DataHub WebView

Check the data domains that you want to access from DataHub WebView. Use the Add... button to create and add new domains.

Start in Run mode

Allows you to start in Run mode, rather than Design mode, with these options:

Kiosk mode

Presents just the working screen of the web browser, with no border, menus, URL entry field, etc. To escape from Kiosk mode (and close the browser), press Alt + F4.

Disable Design mode

Prohibits any switch from Run mode to Design mode, whether running in Kiosk mode or normally.

Disable data writes from client

Prevents the DataHub WebView client from accessing DataHub point values.

Show page information icon

Shows or hides the page information icon.

Load a page at startup

Allows you to specify a page that will automatically load when DataHub WebView starts.

Use a custom branding folder

Allows you to specify a folder for holding custom branding information. For details, please refer to Customizing WebView in the DataHub WebView manual.

Launch DataHub WebView in a browser

Provides a convenient way to start DataHub WebView to check this configuration.
Silverlight Access Policy Server

The Silverlight Access Policy is an XML file used to control cross-domain resource access for HTTP and socket connections for DataHub WebView. The contents and use of this policy are beyond the scope of this documentation.

MQTT Client

The MQTT Client option lets you configure the Cogent DataHub as an MQTT client to any number of MQTT brokers, with pre-configured connections for Azure IoT Hub, Google IoT, and Amazon IoT Core.

The DataHub implements bi-directional data transfer between the MQTT broker and standard industrial protocols. It provides some configurability of the MQTT message format, allowing you to convert directly between MQTT messages and data points. In effect the DataHub can act as an industrial MQTT gateway.

Make the Connection

MQTT Client Connections

The Cogent DataHub can act as a client to one or more MQTT brokers.

Check the Send Data to MQTT Brokers box to enable MQTT client functionality. Since the DataHub can be a client to more than one MQTT broker, you need to specify broker infor-
Properties Window

Information for each MQTT client connection. Once you have a broker listed, you can activate or deactivate the connection using its On check box.

To add a broker, press the Add button to open the Connect to MQTT Broker window described below. To edit a broker connection, double-click it or select it and press the Edit button to open that window. To remove a broker, highlight it and click the Remove button.

The Connect to MQTT Broker window

To define or edit an MQTT broker connection, click the Add or Edit button to open the Connect to MQTT Broker window:

Here you have four initial options: configure a standard MQTT connection, or choose one of Azure IoT Hub, Google IoT Hub, or Amazon IoT Core. Each of these three broker-specific tabs provides entry fields unique to that connection which assist in filling in the configuration explained below.

Connection

To configure a standard MQTT connection, click the Standard MQTT tab and enter the following information:

**Label**

A name used by the Cogent DataHub to identify the connection. There should be no spaces in the name. It doesn't matter what name is chosen, but it should be unique to other connection names.

**Host Name/IP**

The name or IP address of the computer running the MQTT broker you want to connect to.

**Port**

The number of the open port on the MQTT broker that will receive your connection request. The default, 1883, is the port typically used for MQTT.
Keepalive
The number of seconds between MQTT keepalive messages, which are transmitted between the broker and client to help them detect a network failure or lost connection. The default is 30. Lower numbers will detect network failures more quickly, but will also slightly increase message traffic and may result in false failure detections on slow networks.

Retry rate
The number of milliseconds to wait before retrying a failed connection.

Maximum update rate
A limit on how often the DataHub sends packets to the MQTT broker. The default, 0, sets no limit.

The maximum update rate for the broker depends on the **Maximum message length** to some degree. Please see **Broker Limits** for more information.

Authentication
Enter your MQTT broker authentication information, as applicable.

Use SSL
MQTT over SSL typically uses a self-signed certificate, or one certified by an official certificate authority (CA). Here you can enter or browse for one or both of these, as well as enter a password. Checking the **Accept invalid certificates** box will have the DataHub attempt to use SSL without a certificate.

Client ID
An ID that the broker is expecting for this MQTT connection.

Username
The user name for this ID.

Password
The password for this username.
Message Content

MQTT does not have a set format for message content; different MQTT brokers often require different formats. You need to know the format used by the broker you are connecting to.

Message format

Here you can edit the format of your message to match to the format of the broker you are connecting to. Click the Edit Format button to open the Configure Parser window:

The JSON Path fields at the top show your current entry. To make changes, edit the Message Start, Per-Point Format, Per-Point Separator, and the Message End fields. When you press the OK button your changes will be registered. To make a new entry, such as "PointName", enter a string and the placeholder (like {point} in this example) using the same syntax as the other entries in the Per-Point Format entry field.

- The simplest way to pass an MQTT message is with unparsed values. To configure this in the DataHub, the Per-Point Format must be simply:

  ```
  {value}
  ```

  with no entries in the other fields.

- The default Per-Point Format can be used for single DataHub points:
• To collect the data from multiple points and send it as a single message, you can do the following:

1. In the **Connection** settings, change the **Maximum update rate** to a non-zero number to allow the DataHub to batch values by time.

2. In the **Push data point to the MQTT broker** settings, check the box **Send all messages to this topic**, and enter a topic name.

3. In the **Message Content, Message format** settings, use open and closed square brackets ([ and ]) for the **Message Start** and **Message End** entries, and a comma (,) for the **Per-Point Separator**. Then make the following entry for the **Per-Point Format**:

   ```json
   { "TopicName": {topic}, "PointValue": {value}, "PointQuality": {quality}, "PointUnixTimestamp": {unixtimestamp}, "SenderId": {sender} }
   ```

   • White space does not matter.
   • The order of the properties does not matter.
   • Capitalization in property names *does* matter.
   • If topics are encoded in each element of the array (as above), the destination topic to which the client sends this message does not matter. The destination topic must still be well-formed, it just will not get written to.

   These settings will tell the DataHub to collect values for the amount of time specified in the **Maximum update rate**, and will send all point values as a batch to the specified topic.

When pulling data from the broker, the broker's payload format may not map directly to DataHub points. Please see **Message Payload Formats** for information on how to handle this case.

**Quality of service**

MQTT supports 3 levels of quality of service:

• **At Most Once (0)** Every message will be delivered on a best-effort basis, similar to
UDP. If the message is lost in transit for whatever reason, it is abandoned—the receiver never receives it, and the sender does not know that it was lost.

- **At Least Once (1)** Every message will be delivered to a receiver, though sometimes the same message will be delivered two or more times. The receiver may be able to distinguish the duplicates, but perhaps not. The sender is not aware that the receiver received multiple copies of the message.

- **Exactly Once (2)** Every message will be delivered exactly once to the receiver, and the sender will be aware that it was received. Some MQTT brokers and services, such as Azure IoT Hub, Google IoT, and Amazon IoT Core do not support this quality of service, and will simply disconnect when you attempt to send a topic update.

If you are not sure, choose **Exactly Once (2)**, the default.

**Retain messages on broker**

Normally an MQTT message that has no subscribers is simply discarded by the broker. This option tells the broker to keep the last message on this topic even if there are no subscribers.

**Last Will Message**

The Last Will and Testament (LWT) feature of MQTT automatically generates a message to all other clients if a client fails unexpectedly. Here you can configure that message, as well as messages for normal startup and shutdown.

**Topic**

The MQTT topic for the message. Leaving this blank will disable the sending of any message.

**Quality of service**

Choose the MQTT quality of service. Please refer to **Quality of service** (above) for more information.

**Value when starting**

The message you want this client to send on start-up.

**Value when stopping**

The message you want this client to send on normal shut-down.

**Value when connection lost**

The message you want this client to send on any kind of unexpected shut-down or disconnect.
Once you have configured the Connection, Authentication, Message Content, and Last Will Message, we recommend testing the connection by clicking the OK button. This will close the Connect to MQTT Broker and display your entries in the list of configured brokers. The Status should eventually change to Running.

When the connection is tested and working, you are ready to exchange data.

Exchange Data

Once your connection is configured, you can create topics on the MQTT broker using points from the DataHub, and/or request topics from the MQTT broker to create corresponding the DataHub points.

Push data point to the MQTT broker

This mode allows you to select points from the DataHub data set and to transmit them to the MQTT broker. Normally you should select Convert dot to slash to allow the DataHub to convert point names to valid topic names. You may also choose Also subscribe to changes in the broker. This will cause the DataHub to listen for changes in the topics and write the results back to DataHub points. Not all brokers support subscriptions.

Choose the points from the Available Points list and they will appear in the Selected Points list. Use the Remove button to remove a selection.

The following options apply to all points selected:

Add prefix segment to every topic
A string gets inserted at the beginning of the DataHub point name, before the domain name. This is useful for creating a single tree for points from different domains. For example, an entry of MyTree here would change a point named DataPid:PID1.Mv to MyTree:DataPid.PID1.Mv. A point named DataSim:Ramp would
become MyTree:DataSim.Ramp. If the **Convert slash to dot** option (see below) is checked, these points would be represented on the MQTT broker as MyTree/Data-aPid/PID1/Mv and MyTree/DataSim/Ramp.

**Send all messages to this topic**

If the broker you are connecting to does not expect a topic for each point, you can specify one destination topic for all points here.

**Convert slash to dot**

The MQTT topics use a slash ( / ) character in hierarchical names, while OPC and other industrial protocols typically use a dot ( . ). Keep this option checked to have the DataHub convert DataHub points into MQTT topics.

**Automatically create point hierarchy**

The MQTT protocol supports hierarchical names. Checking this option preserves the hierarchy within the DataHub.

**Also subscribe to changes in the broker**

Allow the broker to write back to these points in the DataHub.

**Broker Limits**

If your broker has limits on subscriptions, messages, or topic names, you can enter them here.

The maximum update rate for the broker depends on the **Maximum message length** to some degree. If, by the time the DataHub reaches the configured **Maximum update rate** time, it has accumulated more data than can be transmitted due to a **Maximum message length** setting here, then the DataHub will attempt to write as many messages as required to consume all waiting data without exceeding this limit. This will result in messages being sent faster than the **Maximum update rate** setting.

Click the **OK** button to save your changes, or **Cancel** to cancel them.

**Pull topics from the MQTT broker**

This mode allows you to specify topics in the broker that the DataHub should subscribe to. This will cause values in the broker to be written to DataHub points. You may also
choose **Also publish changes here to the broker**. This will cause the DataHub to write values for the selected points back to the broker, if the broker supports this.

Click the **Add** button to enter a point in a pop-up dialog. It will appear in the **Selected Points** list. Use the **Edit** button to make changes, and the **Remove** button to remove a selection. The topics should be supplied in MQTT syntax, like this: `plant1/mixer/motor/speed`. You may specify MQTT topics with the wild cards `+` and `#`.

- The `+` character is an internal wild card that matches any string between `/` delimiters. For example, you can use `plant1/+/motor/speed` to subscribe to all motor speeds in plant1. The topic name cannot start or end with a `+`.
- The `#` character is a terminal wild card that can only appear at the end of the topic, such as `plant1/mixer/#` to subscribe to all topics related to `plant1/mixer`. You may specify a topic that is just a single `#` to subscribe to all topics in the broker.

The following options apply to all topics:

**Place all data points into this data domain**

This is a convenient way to organize your incoming MQTT connections. Choose any existing DataHub domain, or enter a new name and a new domain will get created in the DataHub.

**Convert slash to dot**

The MQTT protocol typically uses a slash `/` character in hierarchical names, while OPC and other industrial protocols often use a dot `.`. Checking this option lets you convert MQTT-style names.

**Automatically create point hierarchy**

The MQTT protocol supports hierarchical names. Checking this option preserves that hierarchy within the DataHub.

**Also publish changes here to the broker**

Allows the DataHub to write values to these points in the broker.

**Broker Limits**

If your broker has limits on subscriptions, messages, or topic names, you can enter
them here.

Click the **OK** button to save your changes, or **Cancel** to cancel them.

**Pre-Configured Connections**

Because MQTT is a messaging protocol, not a data communications protocol, it does not specify a particular format for making a connection or the data payload. Thus, each MQTT implementation can be different with its own, unique connection characteristics.

Our **Standard MQTT** option provides a generic way to configure a connection to any MQTT broker. In addition, we offer the following pre-configured options to facilitate connecting to Azure, Google, or Amazon MQTT brokers.

**Azure IoT Hub**

To make a connection to Azure IoT Hub you will need to follow some extra steps. A password must be generated by a separate tool as described in the section called “Azure IoT Password Creation”

Once you have a password, click the **Azure IoT Hub** radio button and enter the following information:

**IoT Hub Name**

The IoT Hub Name provided by Azure.

**Device Name**

A name for the device that you want to connect.
Subscribe to cloud-to-device events

Optionally, you can enter the MQTT message format to receive data and event updates from the IoT Hub.

After making your entries, press the Reconfigure button to add that configuration to the list, and clear the fields for another entry. You will see the necessary information entered in the Connection, Authentication, and Message Content tabs. You can optionally configure a Last Will Message, if desired.

The Reconfigure button also sets the DataHub’s Broker Limits configuration to the default values for this cloud service's broker. If any broker limits are non-zero, (such as if they were previously configured), they will not be reset. To reset them, first manually change them to zero and then click this button again.

When you are finished, you can configure your Exchange Data options.

Azure IoT Hub does not allow a client application to subscribe to a topic. That is the way Azure is designed. It was never intended to be a general-purpose MQTT broker. The only way to get data from Azure via MQTT is through a mechanism called “Cloud-to-device events”. There is a setting in the DataHub configuration for that. You need to generate the events from the Azure portal. The Azure documentation can provide more information: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messaging.

Azure IoT Password Creation

The Azure IoT password is an SAS token. To generate the SAS token:

1. Open your IoT Hub in the Azure portal.
2. On the left menu under Settings find the Shared Access Policies, and click on the policy you want to use.
   On the right you will see the Shared Access Keys.
3. Find and make a copy of the Connection-String Primary. You will need this in the Azure IoT Explorer.
4. Download the current release of the Azure IoT Explorer from: https://github.com/Azure/azure-iot-explorer/releases
5. Install it and run the application.
6. Paste in the copy you made of the IoT Hub Connection String Primary, and click Connect.
   This will list your IoT Devices.
7. Click your device.
8. Find the Connection string with SAS Token section.
9. Set **Symmetric Key** to **Primary Key**.

10. Set the keys expiration time in minutes.

11. Click **Generate**. The result will look like this:

```
HostName=SWTBOPCUAHub.azure-devices.net;DeviceId=TestDevice2;
SharedAccessSignature=SharedAccessSignature sr=SWTBOPCUAHub.azure-devices.net%2Fdevices%2FTestDevice2&sig=FLwubhFB4V%2F7j6pZS3KXEomL4%2F2uCaBSipyKiIZCWuw%3D&se=1592419771
```

(But all one string, no carriage returns.)

12. Copy everything after the part of the string "SharedAccessSignature=". The result should be a single string, similar to this:

```
SharedAccessSignature sr=SWTBOPCUAHub.azure-devices.net%2Fdevices%2FTestDevice2&sig=FLwubhFB4V%2F7j6pZS3KXEomL4%2F2uBaCSipyKiIZCWuw%3D&se=1592419771
```

(All one string, no carriage returns.)

This is the SAS token, which you can paste verbatim into the DataHub configuration **Password** field.

**Google IoT**

Click the **Google IoT** tab and enter the following information as provided by Google:

**Project ID**
- The ID for your project.

**Registry ID**
- Your Registry ID.

**Device ID**
- The Device ID for this device.

**Cloud Region**
- The name of the Google cloud server region.

**Client Certificate**
- The certificate for this connection.
After making your entries, press the **Reconfigure** button to add that configuration to the list, and clear the fields for another entry. You will see the necessary information entered in the **Connection**, **Authentication**, and **Message Content** tabs. You can optionally configure a **Last Will Message**, if desired.

The **Reconfigure** button also sets the DataHub's **Broker Limits** configuration to the default values for this cloud service's broker. If any broker limits are non-zero, (such as if they were previously configured), they will not be reset. To reset them, first manually change them to zero and then click this button again.

You are now ready to configure your **Exchange Data** options.

**Amazon IoT Core**

Click the **Amazon IoT Core** tab and follow the instructions given in the interface:

- **Client Certificate**, **Client Private Key**, and **CA Root Certificate** entries correspond to files that Amazon provides. After entering the correct information for all three fields, press **Reconfigure** and the DataHub will automatically create a certificate file and fill in the members of the **Authentication** and **Message Content** tabs.

The **Reconfigure** button also sets the DataHub's **Broker Limits** configuration to the default values for this cloud service's broker. If any broker limits are non-zero, (such as if they were previously configured), they will not be reset. To reset them, first manually change them to zero and then click this button again.

If you need to change any auto-generated settings, please keep in mind that Amazon IoT Core does not accept connections with quality of service 2, or with message retention, and only allows the following settings in the **Message Content** tab:

- **Retain messages on broker** = not checked
- **Quality of service** = **At Most Once** or **At Least Once**
When you are finished, you can configure your Exchange Data options.

**MQTT Broker**

The MQTT Broker option lets you configure the Cogent DataHub as an aggregation and distribution point for MQTT messages from devices and MQTT clients. It can operate as a standard MQTT broker, or as a gateway broker to the DataHub's data set.

These are the configurable options for the MQTT Broker:

![MQTT Broker Configuration](image)

**Enable MQTT Broker**

This box enables or disables all MQTT Broker functionality.

**Plain text connections**

Check the box to enable plain text connections, and specify a port. Port 1883 is the MQTT default for plain text.

**SSL connections**

Check the box to enable SSL connections, and specify a port. Port 8883 is the MQTT default for SSL. If you are using a certificate for SSL connections, you can enter it here.

**Message Content**

Here you can set the DataHub to act as a standard MQTT broker or a gateway broker.

1. **Standard MQTT broker** In this mode the DataHub does not interpret MQTT messages and simply distributes them to its clients according to standard MQTT protocol. To select this mode, enable the option **Do not process messages, just route them.**

![Message Content](image)
2. **Gateway broker** In this mode the DataHub still performs the role of a standard MQTT broker, while at the same time it interprets the incoming messages, mapping them to DataHub points. Messages from clients produce data changes in all configured DataHub protocols. Changes to data in the DataHub result in messages to MQTT subscribers. MQTT clients can participate in control systems as peers with OPC, DDE and DHTP clients and server.

Messages can be interpreted in three ways:

- **Treat messages as binary** Messages are Base-64 encoded and stored in the DataHub as strings. They are not processed beyond that. Use this mode if you want the DataHub to put the values into data points based on the MQTT topic name. For example, you could use this mode when the MQTT data is an image.

- **Treat messages as text** Messages are interpreted as UTF-8 text and added to the DataHub as point values. No further interpretation is done. Use this mode when the MQTT messages are free-form text or you do not want them to be interpreted further.

  You can use this option to discover the client message format. It will cause the JSON parser to be bypassed, and the client messages to be written as text into the destination topic.

- **Interpret messages as JSON data point values** Messages are expected to be text in JSON format. The DataHub can interpret the messages according to a message format you specify. To edit the message format, click the **Edit** button, which opens the Configure Parser window:

![Configure Parser Window](image)

The JSON Path fields at the top show your current entry. To make changes, edit the
Message Start, Per-Point Format, Per-Point Separator, and the Message End fields. When you press the OK button your changes will be registered. To make a new entry, such as "PointName", enter a string and the placeholder (like {point} in this example), using the same syntax as the other entries in the Per-Point Format entry field.

The Max message length field allows you to set a maximum number of bytes that the broker will send in an individual message. This setting is only relevant when the broker is interpreting messages, and the message format would allow for multiple point values to be combined into a single MQTT message. This setting will limit the number of point values that will be packed into each message. Do not set this value to be smaller than the expected length of a message containing only one point value.

Unparsed Values

The simplest way to pass an MQTT message is with unparsed values. To configure this in the DataHub, the message format must be simply {value}.

To make this entry, click the Edit button to open the Configure Parser dialog, and ensure that the entry for Per-Point Format is only the string {value} with no extra characters. Also, there should be no entries in the Message Start, Per-Point Separator, and Message End fields.

To see how the {value} message format can be used to connect an MQTT client whose message format you do not know, please refer to the section called “Connecting MQTT Clients with the DataHub MQTT Broker”.
Point Names

Place all points in this data domain
You can specify a DataHub domain in which all points related to MQTT topics will be collected.

Convert slash to dot
The MQTT protocol typically uses a slash (\/) character in hierarchical names, while OPC and other industrial protocols often use a dot (\.). This option will convert MQTT-style names.

Automatically create point hierarchy
The MQTT protocol supports hierarchical names. This option preserves the hierarchy within the DataHub.

Options

Require authentication
Enables the MQTT Broker’s support for username and password authentication. The usernames and passwords for this are created in the DataHub Security option, following the Common Scenario recommendations for TCP or tunnel/mirror connections. The MQTT Broker does not check for Read and Write permissions. Any user with Connect permission can both read and write data on the MQTT Broker.

Re-publish current value after a client subscribes
Normally when a client subscribes to a topic in the broker, only that client will receive a notification of the topic’s current value. This setting will tell the broker to generate a new “publish” message to all subscribed clients whenever any client creates a subscription.

Mark data as Not Connected when client disconnects
When the DataHub broker is running in gateway mode it writes the values of MQTT messages into DataHub data points, with Good quality. If an MQTT client disconnects from the broker then this option will cause the broker to change the qualities of every point originating from that client to Not Connected, indicating that the client is no longer connected and the value of the data point is not necessarily up to date. This option applies only to those points whose values were updated by the client during the time it was connected.
Overflow Management

When the broker receives messages more quickly than clients can accept them, it stores the messages in queues, one per client. Here you can manage overflows in those queues.

Maximum per-client message queue length

Set a limit for the maximum number of outstanding messages that are queued per client.

Overflow handling strategy

There are two options for handling a queue overflow:

- **Drop newest message** keeps the older undelivered messages in the queue, and drops newer messages without queueing or transmitting them.
- **Drop oldest queued message** continually updates the queue with the newest data, removing the oldest undelivered messages to maintain the maximum queue length. This is the recommended setting for real-time systems where the most recent data is most important.

Neither of these cases will prevent loss of information if the queue becomes full, and neither case will ensure eventual consistency between the broker and its clients. To avoid data loss, you must ensure that the data rate to the client does not exhaust the client’s ability to consume the data and does not exhaust the available network bandwidth.

If you need a consistency guarantee, consider using a DataHub tunnel instead of MQTT for that communication.

Modbus

The Modbus option provides Modbus master support for Modbus/TCP. Using this with the OPC DA option enables the DataHub to function as a Modbus OPC server.
The **Add Slave**, **Edit Slave**, and **Remove Slave** buttons allow you to add, edit, or remove a slave connection, as described below. The **Load Modbus Configuration** button allows you to load an XML file to configure the DataHub. For an example of the XML code, you can refer to the plugin_Modbus.cfg file located with the configuration files.

Clicking **Add Slave** or **Edit Slave** will open the Connect to Modbus Slave dialog with the following options:

**Connection**

**Name**
A unique name for this connection. The system auto-generates the name MBnnn where nnn is an incrementing number starting at 000.

**Host / IP Address**
The name or IP address of the Modbus slave.

**Port**
The port number used to connect to the Modbus slave. The default port is 502.

**Retry Rate (ms)**
A value in milliseconds of how often the DataHub should attempt to retry the connection to the Modbus slave, in case of failure to connect.

**Share existing socket**
When more than one connection is made to the same IP address and port number there is an opportunity for the DataHub Modbus driver to share the same socket for
all such slave connections. If you select this option, the Modbus driver will find any other slave connections to the same IP address and port, and combine them into a single socket connection. If you do not select this option then the connection to the slave will occur on a distinct socket from other slaves on the same IP address and port. This selection is on a per-slave basis. There can be at most one shared socket to any IP address and port, and any number of non-shared sockets.

**Serialize messages on a socket**

If a socket is being shared, setting this option tells the DataHub Modbus driver to serialize the communication with the devices. That is, not send a message until the previous message has produced a response. This is necessary for some devices that expect the Modbus master to transmit a single message and then wait for either a response or a timeout. For this type of device, if the master transmits more than one message at a time, only the first message is processed and the others are lost.

If none of the connected devices require serialized messages, you should not set this option. It is more efficient to transmit requests to all of the slave devices at once, and then to wait asynchronously for the responses to come back. That way, if a particular slave is off-line or is very slow, it will not affect the timing of the communication with the other slave devices, since the results of the query can arrive in any order and independently of one another. You can think of this kind of communication as parallel multiplexing of the shared socket, instead of serialized messages.

Serializing messages will, in general, produce longer latencies when communicating with more than one slave device, and will introduce significant delays for all slaves when any slave is slow to respond. Effectively, this will couple the slave device polling timing. Thus, you should not set this option unless your target slave device requires it.

If you have configured more than one slave connection to share a socket, then setting this option on any of those slave connections will cause all slave connections sharing the same socket to also be serialized.

**Modbus Options**

The DataHub supports nine of the most commonly used Modbus read and write functions, as illustrated in the following two tables:

**Table 1. Supported Modbus Read Functions**

<table>
<thead>
<tr>
<th>Code</th>
<th>Function Type</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Internal Bits or Physical Coils</td>
<td>Read Coils</td>
</tr>
</tbody>
</table>

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
### Table 2. Supported Modbus Write Functions - Optional

<table>
<thead>
<tr>
<th>Code</th>
<th>Function Type</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Internal Bits or Physical Coils</td>
<td>Write Single Coil</td>
</tr>
<tr>
<td>6</td>
<td>Int/Physical Output Registers</td>
<td>Write Single Register</td>
</tr>
<tr>
<td>15</td>
<td>Internal Bits or Physical Coils</td>
<td>Write Multiple Coils</td>
</tr>
<tr>
<td>16</td>
<td>Int/Physical Output Registers</td>
<td>Write Multiple Registers</td>
</tr>
<tr>
<td>22</td>
<td>Int/Physical Output Registers</td>
<td>Mask Write Register</td>
</tr>
</tbody>
</table>

The Modbus driver can write all of the value types that it can read. You may select which Modbus functions will be available for writing during configuration of the slave connection. If no function is enabled that would allow the driver to write a particular value then that value will not be written to the Modbus slave. The slave device must support Masked Write Register (function 22) in order to write to a bit field within a register.

**Polling Rate (ms)**

A value in milliseconds of how often the DataHub should poll the connection to the Modbus slave for data updates.

**Max message length (bytes)**

The maximum message length that the slave device can support. The Modbus specifications set this length to 256, but some devices may vary from that. Check your device documentation, and set this value accordingly.

**Slave ID**

A value in the range of 0 - 256. This setting is useful for TCP Modbus when the connection goes via a serial gateway to multiple slave devices, and the target slave device needs to be identified. Setting this value to 0 means a non-specific broadcast to all devices.

**Timeout (ms)**

A value in milliseconds for the response timeout.

**Addressing**

These options are provided to facilitate configuration, allowing the selected addresses to match the documentation of the slave device.
Bits within registers start at 1 instead of 0

If you are accessing bit fields within a register, this option lets you choose which bit to start with.

Bit 0 in register is most significant bit

If you are accessing bit fields within a register, this option tells the driver whether to treat the lowest order or highest order bit as the first bit.

When packing digital values into an integer (see below), the order of the digital bits will be determined by this setting. If checked, the highest address in the field will be packed into integer bit 0. If not, the digital bit in the lowest address in the field will be packed into integer bit 0.

Addresses start at 1 instead of 0

A convenience option allowing you to start your address numbering at 1 instead of 0.

Data Points

Here you configure the points in the DataHub that correspond to register addresses in the Modbus slave. Detailed information about the configured points for this slave gets displayed in the list.

Choose Add Point to add a single point, or Add Range to add a range of points.

Data Domain

The data domain in the Cogent DataHub for all of the data points for this slave connection.

Address and Type

The Add Point, Add Range and Edit buttons in the Connect to a Modbus Slave window (above) all open the Configure Data Point Address dialog:
I/O Type
A pick list of the four supported I/O types:

0: Digital Output (coil)
1: Digital Input (discreet)
3: Analog Input (input register)
4: Analog Output (holding register)

When one of these is chosen, the following selections are given default values.

Number Type
Available for analog inputs and outputs only, the options are Integer, Float, and String.

Encoding
Encoding for integers can be 16, 32, or 64 bit; for floats 32 or 64 bit.

Strings are stored as groups of contiguous registers. The bytes of each register in the string are ordered according to the byte order selection (see Swap, below) and then converted to a character string using one of the following encoding methods:

- **ASCII** Each byte is interpreted as an 8-bit ASCII character. Two characters are stored in each register. If the configured number of registers is insufficient to store the entire string then the string will be truncated.

- **UTF-8** Each byte in a register is interpreted as a single byte in the UTF-8 stream. If the string contains only 7-bit ASCII characters then UTF-8 is equivalent to ASCII. A single UTF-8 character can require up to 5 bytes (2.5 registers) to store. The number of characters that can be stored in a sequence of registers is therefore variable. If the configured number of registers is insufficient to hold the entire UTF-8 sequence then all bytes beyond the register length will be truncated to the nearest character. This option is displayed in Flags in the Connect to Modbus Slave window as "u".

- **UTF-16** Each character is normally stored in a single register. For some characters, a second register may be necessary to store the entire character. If the configured number of registers is insufficient to hold the entire UTF-16 sequence then the sequence will be truncated to the nearest character.

Swap
These options allow you to specify the byte order of the data stored in the Modbus slave device. The natural byte order in Modbus is big-endian, meaning that the most significant byte is stored first. The DataHub's Modbus driver assumes that the data
in the slave device is big-endian unless you specify otherwise. If you know that your slave device stores its data in a different order, you can use the swap options to specify which bytes, words or dwords are swapped in the slave device relative to the standard big-endian representation.

All possible byte order combinations can be produced by selecting zero or more of the following:

- **Bytes**: each pair of bytes gets swapped. For example, AB would become BA; ABCD would become BADC.

- **Words**: each pair of words gets swapped. This is only applicable when the length of the value is at least 4 bytes. For example, ABCD would become CDAB; ABCDEFGH would become CDABGHEF.

- **DWords**: two dwords get swapped. This is only applicable when the length of the value is 8 bytes. For example, ABCDEFGH would become EFGHABCD.

These options are displayed in **Flags** in the Connect to Modbus Slave window as "b" for Bytes, "w" for Words, and "d" for DWords.

Other examples:

- Swapping both bytes and words: ABCD - DCBA
- Swapping both bytes and dwords: ABCDEFGH - FEGHBADC
- Swapping both words and dwords: ABCDEFGH - GHEFCDAB
- Swapping bytes, words, and dwords: ABCDEFGH - HGFEDCBA

**Sign**

The sign indicates whether a particular integer should be interpreted as signed or unsigned. There is no distinction in the Modbus slave register, so the sign is applied by the Modbus master driver. The **Signed** option is displayed in **Flags** in the Connect to Modbus Slave window as "-".

**Deadband**

An optional means to filter out insignificant value changes.

| Deadband: | 0 |

When the DataHub Modbus driver writes a value from the Modbus slave device to the DataHub, it notes that value as significant. Each subsequent incoming value gets compared to that last significant value, and is only written to the DataHub when the difference is greater than the amount specified by the deadband. Any new value that gets written then becomes the latest significant value to be used for future deadband comparisons.

An entry of 0 means that no deadband applies, and all value changes will be considered significant.

No deadband is applied when writing values from the DataHub to the
Modbus slave device. All value changes will be written to the device.

**Address**
The Modbus register address(es) to link to DataHub point(s).

The information in the table below also pops up when you hover your mouse over this entry field.

Addresses can be specified as individual values, ranges of values, bit fields within a register, and ranges of bits within a register. Here is a guide of the address formats.

<table>
<thead>
<tr>
<th>Address Format</th>
<th>Description</th>
<th>Example</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Digital Points</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td>A single value.</td>
<td>12</td>
<td>The digital value in address 12.</td>
</tr>
<tr>
<td>address-address</td>
<td>A &quot;packed&quot; integer composed of up to 64 consecutive digital bits.</td>
<td>12-17</td>
<td>A 6-bit integer created by packing digital values 12-17 into integer bits 0-5.</td>
</tr>
<tr>
<td><strong>Analog Points</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td>A single value.</td>
<td>4</td>
<td>The 16, 32 or 64 bit integer starting at register 4.</td>
</tr>
<tr>
<td>address.bit</td>
<td>A single bit within an integer.</td>
<td>4.3</td>
<td>The third bit of the integer starting at register 4.</td>
</tr>
<tr>
<td>address.bit-bit-tend</td>
<td>A range of bits within an integer.</td>
<td>4.3-27</td>
<td>Bits 3 through 27 (inclusive) within the integer starting at register 4.</td>
</tr>
<tr>
<td>address[N]</td>
<td>An array of N consecutive values starting from address.</td>
<td>10[5]</td>
<td>An array of 5 analog values (16, 32 or 64 bit) starting at address 10.</td>
</tr>
<tr>
<td><strong>Strings</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>address[N]</td>
<td>A string consuming at most N 16-bit registers.</td>
<td>10[20]</td>
<td>A string of up to 20 registers, starting at address 10.</td>
</tr>
<tr>
<td>Address Format</td>
<td>Description</td>
<td>Example</td>
<td>Refers to</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>This could hold 40 ASCII characters, between 8 and 40 UTF-8 characters, or between 10 and 20 UTF-16 characters.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**For Digital and Analog Points**

- **address** is the Modbus register address corresponding to the data point, starting from 0 (or optionally from 1), within the I/O block for that register type. I/O blocks can be one of Analog Input, Analog Output, Digital Input and Digital Output. All values of any bit length other than digital values are considered to be analog. All bit fields within a register are also considered to be analog.

- **bit** is the bit number, starting from zero (or optionally from 1) within an integer data type. The bit number cannot be larger than the number of bits in the selected data type. For example, if the data type is a 32-bit integer then **bit** can be from 0 to 31 (or 1 to 32).

- **endbit** is the last bit in a multi-bit bit field. This can be any number from **bit** + 1 to the highest bit number in the selected data type. If **endbit** is equal to **bit**, it will be treated the same as if **endbit** was not specified. The resulting bit field will be converted to signed or unsigned integer of the same size as the selected integer type.

**Packing Digital Values**

- When packing digital values into an integer, the order of the digital bits will be determined by the setting **Bit 0 in register is most significant bit (see above)**. If this is not checked then the digital bit in the lowest address in the field will be packed into integer bit 0. If this option is checked then the highest address in the field will be packed into integer bit 0.

- Packed digital values can be up to 64 bits in length. You can use the **Convert to** type option in the **Transform** section to determine the data point type of the result. You may specify any signed or unsigned integer type, though you should specify a type that is large enough to hold the selected number of bits.

**For Strings**

- **N** is the number of 16-bit registers to use for storing the string. This is the maximum number of registers in the string, not the maximum number of characters. The maximum number of characters will depend on the character encoding selected, and the string to be stored. For example, UTF-8 strings can require up to 5 bytes per character.

- The bytes in a string are packed into the registers such that 8-bit types (ASCII and UTF-8) store 2 characters per register and UTF-16 stores one character per register.
The Swap option (see above) determines which of the two bytes is treated as the first character.

**Point name**

Any valid DataHub point name, without a domain name. (The domain is specified in the Connect to Modbus Slave dialog). A dot (.) in the point name will create an assembly or sub-assembly in the domain hierarchy.

**Ranges**

A range is a labour-saving method for specifying many similar data points at one time. By choosing Add Range instead of Add Point for adding points, you can specify a point once, and then enter a value in the Item Count to repeat that specification for that number of points. The range will automatically create new points with consecutive addresses based on the selected data type.

Point names are numbered consecutively, starting from the Start Number that you specify, incrementing by one for each point name in the range. The point name in a range must contain a .NET format specifier that tells the driver how to construct the point name and where to insert this number. The specifier is one of these forms:

- `{0}` The current sequence number, with no additional leading zeros.
- `{0:Dn}` The current sequence number, prefixed with additional zero characters to pad the number to n characters.

The format specifier does not need to be at the end of the point name. For example, the Point name entry:

```plaintext
mb.tank{0:D2}_level
```

with an Item Count of 3 and a Start Number of 5 would produce point names like this:

```plaintext
mb.tank05_level
mb.tank06_level
mb.tank07_level
```

If you specify a range of bit masks within registers, then these rules apply:

- If the bit field contains exactly one bit then the range will be all consecutive bits within the same register (or 32- or 64-bit integer as specified in the data type).
- If the bit field contains more than one bit then the range will consist of the same bits taken from consecutive registers (or 32- or 64-bit integers as specified in the data type).
Properties Window

Allow writes to Modbus device
Enables or disables writes to the Modbus slave for this point or range.

Transform

Specify the type of transformation by clicking Direct copy, Linear Transformation, or Linear Range Mapping.

- Direct copy does not apply a transform. It copies the value from the Modbus slave directly to the DataHub point.
- Linear Transformation lets you multiply by one value and add another value, such as in the equation $y = mx + b$ where the DataHub point is $y$, the Modbus value is $x$, the multiply by value is $m$, and the then add value is $b$. For example, to transform a Celsius value in the Modbus slave to a Fahrenheit value in the DataHub, you would multiply by 1.8 and add 32, or
  
  \[
  \text{Fahrenheit} = (1.8 \times \text{Celsius}) + 32
  \]

- Linear Range Mapping lets you enter a range for the Modbus slave and DataHub Data Points, and the DataHub automatically calculates the corresponding linear transformation.
For example, the multiply by and then add entries to convert a Fahrenheit value in the Modbus slave to a Celsius value in the DataHub would not be simple round numbers. But you can let the system do the math by using the entries of 32 and 212 for the Min and Max of the Modbus value, and 0 and 100 for the Min and Max of the Data Point. As soon as you make these entries, the correct values get entered automatically in the Linear Transformation.

Entries for Clamp are applied to all values read into or written out of the DataHub for this point. So, the value read from the Modbus slave will never exceed the clamped range. Or, when a value is written to the DataHub point from another source, the clamp is applied before writing the transformed value to the Modbus slave. It is possible for the value in the Modbus slave to be different from the value in the DataHub if a value beyond the clamped range is written to the Modbus slave by an external program. In that case, the DataHub point will report the clamp value, even though the value in the Modbus slave is out of range.

- **Convert to** lets you convert an input value from the Modbus slave into a signed or unsigned integer, float, or double for the DataHub point value.

  When applying a transformation to a value, the result is frequently a non-integer. Thus, it is a good idea to convert the value to a floating point (float or double) number when using a transform.

  You can use this option to determine the data point type of a packed digital value (see above). You may specify any signed or unsigned integer type, though you should specify a type that is large enough to hold the selected number of bits.

**Historian**

The Historian option allows you to collect and store histories for groups of data points. It gets configured automatically by the Quick Trend option, and can be configured manually as well.
Check the Enable historical data collection box to enable the Historian. Histories for one or more points are created and stored in groups. You can create as many groups as your system resources will allow, and activate the data collection for that group using its corresponding On check box in the list. To edit a history group, double-click it here, or select it and press the Edit button to open the Configure Historical Data Capture window (see below). To remove a group, highlight it and click the Remove button.

**Group Configuration**

To add a history group, press the Add button, which opens the Configure Historical Data Capture window:

**Group label:**
Any text string, used to identify the group. The Cogent DataHub will assign a numbered label by default if nothing else is specified.

**Base directory:**
The directory in which the histories will be stored.

**Change file every:**
How often to close a recorded history file, and open another.

**Keep data for:**
How long to store data on disk.

**Flush to disk every:**
How often to flush the data from memory and write it to disk.

**Example:**
The directory tree and example filename that the history will be written to.

**Deadband**

A deadband is used to reduce the amount of data stored by only storing data if there is a significant change in value. This approach is superior to simply reducing the sampling frequency, which will lose information when data changes quickly, and will waste storage by saving the same values when data doesn't change. The deadband approach defines a resolution below which changes in data are deemed to be 'noise' and therefore ignored.
Absolute change:

Sets the deadband range to a single value. Any new value differing from the baseline (current value) by less than the number entered here will not be recorded. If a value's difference from the baseline is greater than or equal to this number, that value gets recorded and it becomes the new baseline for the absolute change deadband.

Percent change:

Sets the deadband range to be a percent of the last logged data value. Every new value is compared to that value, and if their percentage difference is less than the number entered here, the new value will not be recorded. If the difference is greater than or equal to this number, the value is recorded and that value becomes the new baseline for the percent change deadband.

If absolute change and percent change are used together there is an AND relationship between them. The Historian will ignore any value falling within either deadband. Only those values falling outside all deadbands (or equal to the outermost) will be recorded.

Maximum time between values:

Sets the maximum time period (in seconds, a real number) to deadband. When the maximum time is reached, the next new value received will be recorded, even if its value doesn't exceed the deadband. Note that the system does not automatically generate and insert a value when the maximum time is reached; it waits for the next new value. Whenever a new value is recorded the time calculation is restarted. If the maximum time parameter is not used, there is no time limit on how many values are ignored within a deadband.

Maximum number of skipped values:

Sets a maximum number of values to skip for the deadband. When the maximum number is reached, the next new value will be recorded, even if it doesn't exceed the deadband. Whenever a new value is recorded the countlimit is restarted. If the countlimit parameter is not used, there is no count limit on how many values are ignored within a deadband.
In-Memory Cache

**In-memory cache size (# of values):**

Determines the amount of memory allocated to storing values. This is specified by entering a number of values.

**Select Points Using These Settings**

Any number of points can be selected for the group, using the selection tree.

Click the **OK** button to save the group configuration and return to the Historian section of the Properties window. Then click the **OK** or **Apply** button in the Properties window to ensure t

**Querying Data in the Historian**

It is possible to query the Historian to extract raw data as well as statistics like minimums, maximums, averages, variances, standard deviations and so on over a given time period. This is done with a script that accesses the **Historian class**, explained in detail with an example in the DataHub Scripting manual.

**Historian File Format**

The format for the Historian files consists of consecutive records as follows:

- **timestamp**
  
  An 8-byte floating point number indicating the number of seconds since January 1, 1970 00:00:00, UTC time, in IEEE floating point format.

- **value**
  
  An 8-byte floating point number, in IEEE floating point format.

- **quality**
  
  A 2-byte integer number encoding OPC quality, in little-endian order (Intel byte order):

  ```
  OPC_QUALITY_MASK = 0xc0;
  OPC_STATUS_MASK = 0xfc;
  OPC_LIMIT_MASK = 0x3;
  OPC_QUALITY_BAD = 0;
  OPC_QUALITY_UNCERTAIN = 0x40;
  OPC_QUALITY_GOOD = 0xc0;
  OPC_QUALITY_CONFIG_ERROR = 0x4;
  ```
OPC_QUALITY_NOT_CONNECTED = 0x8;
OPC_QUALITY_DEVICE_FAILURE = 0xc;
OPC_QUALITY_SENSOR_FAILURE = 0x10;
OPC_QUALITY_LAST_KNOWN = 0x14;
OPC_QUALITY_COMM_FAILURE = 0x18;
OPC_QUALITY_OUT_OF_SERVICE = 0x1c;
OPC_QUALITY_WAITING_FOR_INITIAL_DATA = 0x20;
OPC_QUALITY_LAST_USABLE = 0x44;
OPC_QUALITY_SENSOR_CAL = 0x50;
OPC_QUALITY_EGU_EXCEEDED = 0x54;
OPC_QUALITY_SUB_NORMAL = 0x58;
OPC_QUALITY_LOCAL_OVERRIDE = 0xd8;

Email/SMS

The Email/SMS option lets you configure the Cogent DataHub to send emails and SMS text messages. The outgoing mail server is configured once, and each email message is configured separately.

For more information about sending emails and SMS messages, please refer to Email and SMS.

Configure Outgoing Mail Server

SMTP Server:
The name of the SMTP server. You can use the same SMTP server listed in your email client program.

Port (default 25):
The SMTP port number (typically this is port 25).

Sender Email:
The email address of the sender. This will appear in the From field of the email.
User name:
The log-in name you use to access this SMTP account.

Password:
The applicable password.

Security

Never attempt to connect securely via SSL
Ensures that SSL is not used.

Always use SSL (fail if unavailable)
Ensures that SSL is always used, and that any connection attempt without it will fail.

Automatically select most secure connection
Allows the DataHub to choose the most secure type of connection available on the mail server.

Accept invalid or untrusted certificates
Ignores security certificate warnings.

Click the OK button to submit your entries.

Configure Email and SMS Events

To Configure Emails press the Configure button to open the Email/SMS Events window:
A complete explanation for this window can be found in *Email and SMS*, starting with the *Sending a Test Message*.

Click the **OK** button to submit your entries.

**Camera**

The Camera option allows you to stream images and videos from USB cameras, as well as from IP cameras that implement standard JPG and Motion JPG interfaces. This allows the DataHub to collect video from any compliant camera on your network and make the video frames available as data points that can be tunneled and processed by any DataHub client.

[Click here to watch a video.]
Check the Enable Camera box to enable a camera. You can configure as many cameras as your system resources will allow, and activate each one using its corresponding On check box in the list.

To add a new camera, click the Add button. To edit the configuration of an existing camera, double-click it here, or select it and press the Edit button, which opens the Define Camera window (see below). To remove a camera configuration, highlight it and click the Remove button. To view a camera's output, click the View button.

**Define Camera**

To add a new camera, press the Add button, which opens the Define Camera window:

**Camera Name:**
Choose a camera that is available on your system.

**Resolution**
The resolution of the camera image, in pixels, along with the number of frames per second that a video image will be captured.

**Frame Timing (ms)**
The time lag between each frame, in milliseconds.

**Image**

**Scale**
Allows you to scale the original image, according to these parameters:

**Width**
A maximum width, in pixels.

**Height**
A maximum height in pixels.

**Preserve aspect ratio**
Ensure that the image keeps its original aspect ratio, potentially overriding one of the above entries.

**Quality**
The picture quality, expressed as a percent. Where high resolution image quality is not
needed, lowering this setting can significantly conserve system resources.

Point Bindings

Enabling any or all of these point bindings allows you to monitor the camera settings as DataHub points. By checking the **Writable** option, you can make changes to these settings at run time.

![Point Bindings](image)

Data Domain

Allows you to create or specify the DataHub data domain for these points.

Prefix

Allows you to append an identical prefix to the name of each point. This is helpful when multiple cameras have been configured.

On

Turns the point binding on or off.

Property

The camera property associated with the point.

Point Name

The point name for the camera property, which can be edited.

Writable

Determines whether a user can write back to this point to control the camera or modify its settings.

The **Pan**, **Tilt**, **Zoom**, **Roll**, **Exposure**, **Focus**, and **Iris** properties may or may not be supported, depending on the camera hardware.

System Monitor

The System Monitor option allows you to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. with the Cogent DataHub.
For example, by monitoring process ID you could determine whether a particular process is running or not. Any information accessed here becomes part of the DataHub's data set, and can thus be tunneled across the network, used in scripts or as email triggers, viewed in a spreadsheet, or stored in a database.

![System Monitor Configuration](image)

To enable system monitoring, check the **Enable system monitoring** box. There are several configuration options.

**Data Domain:**
The name of any DataHub data domain. The values retrieved from the system will be shown as points in this data domain.

**Update Rate:**
The frequency that the system is polled and all selected points are updated. The minimum polling time is 100 ms., so the value entered here cannot be less than 100.

A high update rate (a low number here) for many data points could use a great deal of CPU.

**When a parameter is invalid**
A parameter will be invalid if the object being monitored is not available. For example, if a process is not running then the parameters for that process will all be invalid. This is a useful way to monitor a system process or other object. For example, you could use a script or other client to watch a process ID, and when the process ID becomes \(-1\) you could generate an alarm indicating that the process is no longer running.

**Point Names**
The System Monitor automatically creates Cogent DataHub point names based on the names of the system properties. Some client programs cannot work with point names containing special characters. This section allows you to specify which characters will be removed from the property name when constructing the point name.
Monitored system parameters

This list shows the system properties you have chosen, and their corresponding point names in the DataHub. To add names to the list, click the Select From List button. This will open the Select System Properties dialog:

Depending on your system, this dialog may take a few seconds to appear. If it does not come up, the Event Log will contain a message. Otherwise, just be patient, it will open eventually.

In the Select System Properties dialog you can specify which items to add to your list of monitored system properties, according to these criteria:
• **Performance object** A list of all available objects, such as CPU, Memory, Process, Print Queue, TCP, etc.

• **Counters** All of the available data categories related to the selected performance object. You can choose all counters, or select specific counters from the list. The **Explain** button opens a window with an explanation of the selected counter.

• **Instances** All of the instances of the chosen performance object. For example, if you chose Process for your performance, this list will show all of the processes running on your system. You can choose all processes or select specific processes from the list.

A number in this list normally indicates a selection from multiple objects of a given type, and _Total means the total across all of the objects. For example, if you are looking at Processor in a multi-processor machine, you will see a number (0, 1, etc.) for each processor and a _Total for the cumulative statistic over all processors.

1. Select a performance object, and counters and instances as applicable.
2. Click the **Add** button to add the selected items to the **Monitor system parameter** list in the DataHub Properties window.
3. Click the **Apply** or **OK** button in the Properties window when you are finished making your choices and filling the list, to apply your changes. You should be able to view the results in the Data Browser.

If you change your mind on what points to monitor, you can change the list at any time. Any points you remove from the list will continue to exist in the DataHub until it is shut down and restarted. Please refer to the section called “Data Points” for more information on creating and deleting points.

---

**DDE**

The DDE option lets you configure the Cogent DataHub to act as a DDE client and/or DDE server for **DDEAdvise** messages. For more information on DDE, please refer to the section called “DDE Protocol” and **Appendix G, DDE Overview**.

For information about connecting to Excel, please refer to **Excel Connections**.

Click here to watch a video.

**DDE Client**

To act as a DDE client, check the **Act as a DDE client** box.

For best performance, ensure that a DDE server is running when using the DataHub as a DDE client. A DDE client can consume substantial system resources trying to connect if a DDE server is not available.
To add a new service or topic, press the Add button. To edit a service/topic, double-click it or select it and press the Edit. To remove a service/topic, select it and press the Remove button.

Double-click a selection, or pressing the Add or Edit button opens the DDE Item Definition Window:

Here is what must be entered:

**Connection Name**
A name used by the Cogent DataHub to identify the connection. There should be no spaces in the name. It doesn't matter what name is chosen, but it should be unique to other connection names. Each connection can have only one Service and one Topic, but it can have multiple Items.

**Service**
The service name of the DDE server to this client.

**Topic**
The DDE topic under which the data will be sent.

**Item Names**
The DDE item name, which corresponds to a point name in the Cogent DataHub. If
that point doesn't exist in the DataHub, it will be created. Enter a name for each item and press the Add button to add items. To remove an item, highlight it and press the Remove button. Once all the information is entered correctly, press the OK button to enter the definition.

DDE Server

To have the Cogent DataHub act as a DDE server, check the Act as a DDE server box. The default service name is datahub.

To add a DDE service name to the list, click the Add button and enter the name in the DDE Service Window:

To edit a DDE service name, double-click it or select it and press the Edit. To remove a DDE service name, select it and press the Remove button.

It is currently possible to have more than one instance of the Cogent DataHub, with one or more DDE service names in common, running on a single machine. If you plan to configure a system like this, you should ensure that each instance of the DataHub has unique data domain names. Otherwise, when any two of those DataHubs are acting as servers, it is not possible to predict which one of them a given client will send data to.

Non-English Characters in Excel

Checking this box will cause Excel to send strings of Unicode characters correctly, although slower than numerical data.

Quick Trend
The QuickTrend option lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.

To configure a quick trend, press the **Open quick trend window** button:

**Configuration**

QuickTrend supports multiple display configurations, which you can manage using the **Configuration** options. The top button opens a dropdown list that contains all named configurations. The **New...**, **Rename...**, and **Delete** buttons let you create, rename, and delete configurations.
Data Points

You can select any number of data points to trend, from any data domain. The following options determine how the data will display in the chart.

Square

Removes interpolation of the line between two data changes, giving the plot a step-like appearance. This is useful for square functions.

Extend

When checked, this option extends the plot of a point's value as a straight line until the point changes value. With this option unchecked, no plot is shown until a point changes value. Then a straight line is plotted connecting the original value to the new value.

Offset

A value entered here will be added to each value of the point, creating an offset plot. This lets you view widely differing values in the same window.

Scale

A value entered here will be multiplied by each value of the point, creating an enhanced (or diminished if a fractional value) plot.

Buffer

This value determines how many data changes for this point will be stored in the local history, to allow for scrollbacks to review recently plotted data.

Label

Allows you to change the label for the point, whose default is the simple point name.

Working with the Display

There are a few features of the display that are not immediately obvious.

• You can scroll backwards and forwards through the history of the trend using the left and right arrow buttons, or choose a specific date and time with the calendar and time selector. The double-right arrow button returns the display to real-time trending.

• You can resize the trend display by dragging the gray borders on the left side or bottom. Move the mouse until you see a white, double-headed arrow, and drag.

• To zoom in to a part of the display, drag the mouse over the area that you want to zoom in on. To zoom back out and resume real-time trending, click on the Reset Pan/Zoom button at the bottom of the window.

• To show a constant value as a horizontal line, you can simply create a DataHub point whose value never changes, create a plot for it, and check the Square and Extend options.

• For other features, the QuickTrend display has a menu available on a right mouse-click.
Using this menu you can copy, save, or print the current display, as well as unzoom and undo a zoom or pan. You can also hide or show point values, and reset the scale to the default.

**Chart**

**Title:**
The name of the chart, which will appear at the top.

**X Axis**
The X axis displays the time coordinate.

**Label:**
Any text string, displays at the bottom of the chart.
**Properties Window**

**Time Span (s):**
The total number of seconds that the chart will span.

**Update (ms):**
The update rate for the trend, in milliseconds.

**Major Tick (s):**
The time interval between major tick marks, in seconds.

**Minor Tick (s):**
The time interval between minor tick marks, in seconds.

**Y Axis**

The Y axis displays the value coordinate.

**Label:**
Any text string, displays at the far left of the chart.

**Minimum:**
The minimum value to include in the chart.

**Maximum:**
The maximum value to include in the chart.

**Major Tick:**
The value between major tick marks.

**Minor Tick:**
The value between minor tick marks.

**E+H Fieldgate**

The E+H Fieldgate option allows you to connect the Cogent DataHub to any number of Endress+Hauser (E+H) Fieldgate servers or RSGx devices.

**Installation**

The E+H Fieldgate feature is an add-on plug-in component for the Cogent DataHub. It must be specified when installing the DataHub. To do so, select it in the optional list of Device Drivers provided during the DataHub installation:
Once installed, the E+H Fieldgate option will be available in the Properties window. Selecting it will open the Endress+Hauser Fieldgate Configuration interface:

**Connection**

**Polling Rate**

Set the number of seconds to poll connections. This is a global rate for all servers.

**Configuration**

To configure a new connection, click the **Add** button. To edit a configured connection, select it and then click the **Edit button**. In either case, this will open the Configure Fieldgate Server dialog:
Address (URL)
The URL for the E+H Fieldgate server or RSGx device. The URL must be the full URL to the XML statistics page on the Fieldgate device. The format will look something like this:

```
http://n.n.n.n:port/xmlpath
```

where `:port` indicates a port other than the default HTTP port (80), and `xmlpath` indicates the path to the XML report from the device. The XML path is normally `index.xml`. You need to include the `http://` at the beginning (or `https://` if it is using HTTPS). For example:

```
http://10.55.22.129:8080/index.xml
```

Username
An alphanumeric string to serve as the user name for this connection.

Password
A password for the user name.

Data Domain
A data domain in the Cogent DataHub where your data will be made available.

Branch
A branch in the data domain, useful for keeping multiple connections separate within a single data domain.

Remote Config
The DataHub installation includes Remote Config, a tool that lets you connect to the DataHub while it is running as a Windows service, or when it is running on a different computer on the network. The Remote Config option here in the Properties window lets you specify what configuration options Remote Config will have access to.
You must configure the DataHub to allow remote configuration for any features that you wish to be able to remotely configure. For example, you may want to remotely configure Tunnel connections, but not allow scripts to be configured except from the local computer.

Check the boxes of the features you want to be configurable. Checking a box in the Local column allows connections only from Remote Config running on the same machine, whereas boxes in the Remote column allow connections from Remote Config running on a remote machine. If you uncheck Local and check Remote, then a user on the local machine will not be able to configure that feature. The Deny All and Allow All buttons uncheck or check all the boxes.

For more information on using Remote Config, please refer to the Remote Config section.

**Scripting**

The Scripting option lets you write, edit, and run scripts, as well as work with configuration files. Please refer to the DataHub Scripting manual for more details.

**Load these Gamma script files when starting:**

Here you can create and access Gamma scripts.
The DataHub Scripting manual has a more complete explanation for these options, but briefly they are as follows:

- **New** creates a new script from parameters you specify.
- **Add** lets you add a script to the list.
- **Edit** opens the script in the Script Log for editing.
- **Load Now** loads the selected script into the Gamma interpreter for immediate processing.
- **Remove** deletes the selected script from the list.
- **Move Up** and **Move Down** move the selected script up and down in the list

**Script Manager**

The DataHub’s Script Application Manager let you view a list of scripts and stop a running script. To access it, press the **Script Manager** button.

This will open the Script Application Manager window:

To stop a script, highlight it, and press the **Stop Selected** button.
The columns display the following information:

- **Class**: the name of the instance of the Application class created in the script.
- **Timers**: the number of timers active in this script.
- **Change Events**: the number of change events active in this script.
- **Menu Bindings**: the number of menu entries that this script has placed in the system tray menu.
- **Sub-Menu Name**: the name of the submenu in the system tray menu into which this script has placed its menu entries.

**Load these additional configuration files when starting:**

Configuration files let you start the Cogent DataHub from a known configuration. Normally there is one configuration file that comes with the DataHub distribution. Each change you make in the Properties window gets written to that file automatically, and saved. However, it is also possible to create and/or edit configuration files with a text editor (see the section called "Configuration Files").

Any additional configuration files must be listed here in order to be read by the Cogent DataHub. The files in the list are used in order from the top down, with the last file taking precedence. That is, if a value is assigned to a variable from within a configuration file, its value will be changed by subsequent assignments, whether they come from within that config file or from any other.

Creating or editing configuration files should only be attempted by experienced users.

The Add button opens a standard file browsing window, from which you can choose configuration files to add. The Remove button removes the configuration file that is highlighted.

The Move Up and Move Down buttons rearrange the order of the configuration files. Order is important. These files are read from the top down; variables change with each subsequent assignment, taking on the last value assigned.

**Security**

The Security option lets you configure security for the Cogent DataHub tunnel/mirror, TCP, OPC, and DDE connections.
For more information on DataHub security, please refer to **Security**.

Click the **Configure Permissions** button to open the Edit Permissions window.

Here you can create and modify groups, and then assign users to those groups.

**Groups**

Groups provide a convenient way to configure a number of users who have identical permissions. Each group can be assigned a unique set of permissions from the **Permissions** table. There are several default groups, such as **BasicConnectivity**, **HTTPUser**, **Admin**, and **RemoteConfig**. To add a group, type a group name in the bottom row of the **Groups** table. Check or uncheck the boxes to assign permissions.

For example, in the illustration below an **Operator** has been added that has been given Connection permissions for **Connect**, **Read**, and **Write**.
Users

There are two kinds of users: normal and special. Normal users correspond to individuals with a name and a password. Special users provide a way to offer different security models for different protocols. For more information on types of users, please refer to the section called “User Authentication”.

To add a user, type a user name in the bottom row of the User table. When you press Enter, a password dialog will appear:
Enter a password and select an HTTP realm for that user. When you click OK, a string of characters will appear in the Password field for that user. Passwords are stored using a reasonably strong non-reversible encryption. If a user forgets his password, it is not recoverable. To change HTTP realm for a user, their password must be reentered. For more information on passwords, please refer to the section called “Passwords”.

HTTP Realms

Here you can maintain a list of HTTP authentication realms. This list is accessed by the DataHub Web Server, as described in the section called “Web Server”.

To add an authentication realm, simply type it into the list. One or more of these realms are assigned to each user when their password is configured (see above).

Common Scenario

The most common Cogent DataHub security configuration is to allow any user to connect via OPC or DDE, while only allowing authorized users to connect via TCP or via a tunnel/mirror. This eliminates exposure of the TCP and tunnel/mirror connections to unwanted Internet and network clients. OPC and DDE are not exposed in this way.

To configure this scenario, you need to remove all group memberships from the special Anonymous, TCP, and Mirror users. Simply click on each of these user names in turn, and uncheck all group memberships for that user. When you are finished, only DDE and OPC should have any group memberships.

Permissions

Connect

This user is allowed to maintain a connection to the Cogent DataHub. When a connec-
tion is made, the client has a 5-second grace period in which to authenticate before the client is disconnected. If the client does not have Connect permissions after the grace period expires, it will be disconnected.

**Read and register points**
This user is allowed to read point values and subscribe to point value changes.

**Change point values**
This user is allowed to write a new point value to the Cogent DataHub.

**Force value changes**
If the user has Change point values permission, he may also have this permission. In this case, the user will be able to send the force and cforce commands to the DataHub, which will override the read-only status and timestamp check for a point, thereby forcing a write to succeed where it would otherwise fail.

**Create new points**
This user is allowed to create new points in existing data domains in the Cogent DataHub.

**Delete an existing point**
This user is allowed to delete a point from the Cogent DataHub.

![Warning]

Normally, no client should be allowed to delete points from the Cogent DataHub. Deleting points can be very disruptive for existing clients. Use this permission with caution.

**Create a new data domain**
This user is allowed to create new data domains. Normally you should also set Create new points permission when you set this permission for a user.

**Load a configuration file**
This user is allowed to tell the Cogent DataHub to load a specific configuration file.

**Create and edit users and groups**
This user is allowed to create and edit users and groups non-interactively.

**Change the program configuration**
This user is allowed to transmit commands to the Cogent DataHub to alter the DataHub's configuration. This normally includes actions like enabling and disabling particular interfaces and functions within the DataHub.

**Change auto domain creation**
This user may change the flag indicating whether the Cogent DataHub should automatically create a data domain when a user requests a point in a non-existent data domain.

**Shut down the program**
This user may transmit an exit command to the Cogent DataHub, causing it to shut down.
The Licenses option lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found on a DataHub running on any connected machine, the Cogent DataHub switches to license mode, and each connected DataHub then requires a license.

This table lists each kind of license, and its current usage.

<table>
<thead>
<tr>
<th>License</th>
<th>In Use</th>
<th>Available</th>
<th>Maximum</th>
<th>Peak Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Link</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Web Data</td>
<td>0</td>
<td>Unlimited</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Excel Plugin</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Admin Link</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MQTT Client Link</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>MQTT Broker Link</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Unlimited DDE</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlimited TCPP</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Details

The Details list shows some of the information embedded in each license string currently installed.
Properties Window

Product Code
One or more codes for the license pack or add-on license product(s) that correspond to this license string.

License #
The license number.

Count
The maximum number of licenses available in this license string.

Type
Either node (unlimited) or connection (a per-connection license).

Expiry
If this is a time-limited demo license, the date on which it expires, or never for a permanent license.

There are two general viewing options for licenses:

Hide icons for unlicensed components
Completely removes the greyed-out property icons for DataHub features that are not licensed.

Show unused license keys
It is possible that a license file may contain upgrade licenses for several different versions or instances of a DataHub. If one or more license files have been loaded, checking this box will show in the Details list all the licenses available, including those that might be for older versions or other installations of the DataHub.

Installing Licenses
Licenses can be entered individually or loaded from a file.

- The Enter a License... button opens the Enter License String window:
Here you can paste or manually enter the text string for the license provided by Cogent. Make sure to include all colon (:) characters in the string.

The license string may contain the characters 1 (lower case L) and 1 (one) which can look nearly identical in some type fonts. If possible, it is best to copy and paste the string, rather than retyping it.

• The **Load License File...** button opens a Windows file selection window. Browse to find the directory and license file that you want to load. License files end with a `.lic` extension. Once you have found the license file, click the **Open** button to load the file. (Please refer to **Configuration and License File Locations** in the section called “Configuration Files” for more information on license file locations.)

To remove a license from your system, select one or more licenses in the **Details** window, then click the **Remove Selected** button.
Other Windows and Programs

Data Browser

This window gives a real-time view into the Cogent DataHub. You can open this window in either of two ways:

- Click the View Data button in the Properties window.
- Right click the DataHub icon in the system tray and select View Data from the pop-up menu.

All data domains in the DataHub are shown in the tree on the left, and all the points of the selected data domain are listed on the right. Clicking on a point name selects it, and puts its name into the Selected Point field at the top of the window. A snapshot of the value of the point at the time you clicked appears in the Enter new value field. You can change the value of the point by entering a new value in this field (i.e. type in the value and press the Enter key). You can also drag the column headers to change the order of the columns.

Drag and Drop Style and Property

There are several options for the drag and drop style, depending on the program in which you want to place the data. In addition, there is a Snapshot (Plain Text) option that allows you to put in labels and data properties that don't change.

The Data Browser also provides a way to select the Property of a data point to drag and drop, such as timestamp, quality, and point type.
The following properties are available:

**Name**
The name of the point, including its data hierarchy of assemblies, subassemblies, attributes, and properties, if any. For example:

```
PID1.Range.Amplitude
```

**Name (Full)**
The name of the point, including its data hierarchy and also domain name. For example:

```
DataPid:PID1.Range.Amplitude
```

**Name (Short)**
The short name of the point, without any of the data hierarchy. For example:

```
Amplitude
```

**Quality**
The quality of the point, an integer.

**Quality Name**
A text string that corresponds to the integer value of the quality of the point.

**Time Stamp (Local)**
The local time stamp, in seconds. In Excel, you can format this using a custom format for the cell. For example, to display the data and time to the nearest millisecond in a worksheet cell, you can use `m/d/yyyy h:mm:ss.000` as your custom format. Note that the milliseconds use a dot, not a colon.

**Time Stamp (UTC)**
The UTC time stamp, in seconds. You can format this in Excel using a custom format, as explained above.

**Type**
The type of the data contained in the point, as an integer.

**Type Name**
A text string that corresponds to the integer value of the data type.

**Value**
The value of the point.
Connection Viewer

This window gives a real-time view into all Cogent DataHub connections. You can open this window by clicking on the View Connections button in the Properties window.

The various columns identify the connection and show its status, with connection statistics, as follows:

- **Sent** Point changes sent from the DataHub to the connecting program.
- **Received** Point changes sent from the connecting program to the DataHub.
- **Created** Data points created in the DataHub by this connection.
- **Registered** Data points registered in the DataHub by this connection.
- **Unregistered** Data points unregistered in the DataHub by this connection.
- **Dropped** Data point changes that the Cogent DataHub attempted to send, but could not, because the client was too busy to receive them before the next point change. When the DataHub drops a point change, it only drops values that have already been superseded by a newer value. The Cogent DataHub will never drop the latest value.
- **Blocked and Unblocked** If the Cogent DataHub identifies that the client is unable to keep up with the transmission data rate, it will mark the client as *blocked*, indicating that the client will receive no new value changes until it is able to cope with them. The DataHub *unblocks* the client after a short period of time, or when it identifies that the client has queue space to accept new data changes.

If a client is blocked, superseded point changes will be dropped. The latest value for each point is queued so that when the client is unblocked it will receive the latest values for all points that changed while it was blocked. Again, the Cogent DataHub never drops the latest value for any point in which a client is interested.

- **CPU** The CPU time in seconds that a given thread has consumed since it started. When the Performance option is selected, the CPU percentage is the average percentage of
the CPU resources that the thread has been using since the Connection Viewer window was opened, or the Set Checkpoint button was pressed.

You have the following options for viewing the data:

- **Total** displays all connection statistics from the time that the client connection was first made.
- **Checkpoint** displays all connection statistics from the last time the Set Checkpoint button was clicked or from the time that the client started, whichever is later.
- **Performance** shows the performance of the connection, in terms of data changes per second, from the last time the Set Checkpoint button was clicked or from the time that the client started, whichever is later.
- **Set Checkpoint** records the time and current statistics for each client, for use by the Checkpoint and Performance options.

### Event Log

This window lets you view a log of events from the Cogent DataHub. You can open this window in either of two ways:

- Click the **Event Log** button in the Properties window.
- Right click the DataHub icon in the system tray and select **View Event Log** from the pop-up menu.
You can specify the category of event **Facility** and level of **Severity** by checking or unchecking the related items. It is possible to open multiple **Event Log** windows, and select different **Facility** and **Severity** options in each window.

The **Debug** option is very verbose. Operating in this mode could put an unusually high demand on system resources.

### Log to File

If you wish to save a log, you can have the Cogent DataHub write the log to a file. Just check the **Log to File** box and specify the file name in the entry field, typically with a `.log` extension. You can use the **Browse** button to find a file. The **Limit** text entry box allows you to change the maximum file size (default 102,400 Kb).

Once the `.log` file reaches the maximum size limit shown in the **Limit** field, it is renamed with a `.log.1` extension, and a new `.log` file is started. Combined, the two files provide at least as much data for analysis as specified in the **Limit** field. This also means that the actual size on disk of the two log files could be as much as twice the **Limit** size at any given time.

Here are some other useful pointers about log file management:

- If the file you specify doesn't exist, the DataHub will create it. If the file does exist, the DataHub will append log data to the file.
- As long as the **Log to File** box is checked, the DataHub will append all log entries to the file, even when restarted after a shutdown.
- You can specify the category of event **Facility** and level of **Severity** logged to file by checking or unchecking the related items.

### Script Editor

The Script Editor\(^1\) lets you write and edit scripts.

\(^1\)This editor is based on the Scintilla and SciTE editor.
The Script Editor offers features such as context-sensitive highlighting, prompted fill-ins for functions and variable names, automatic indenting, text string searches, and so on. Please refer to the DataHub Scripting manual for detailed information.

**Script Log**

This window lets you view output from scripts, and interact with the Gamma scripting engine. You can open this window in either of two ways:

- Click the **Script Log** button in the Properties window.
- Right click the DataHub icon in the system tray and select **View Script Log** from the pop-up menu.
Please refer to the DataHub Scripting manual for detailed information about the Script Log.

DataSim - a data simulation program

DataSim is a data simulation program that creates local data for the Cogent DataHub. It generates data for four different wave patterns, and sends these to the DataHub by a TCP connection.

As soon as DataSim starts, it attempts to connect to a DataHub and begins generating data. To receive the data, the Cogent DataHub should be set up as a tunnelling/mirroring master.

Connection Parameters

Host
The name or IP address of the host computer. Since DataSim connects via TCP, this can be any computer on the network running a Cogent DataHub acting as a tunnelling/mirroring master.

Service
The port number or service name as entered in the Master service/port entry box of the DataHub.

Status
Displays the attempts to connect, which change to Connected when the connection is made.

You can use the Reconnect button to reconnect and the Pause button to freeze all data generation. Press the Show button to view and change some of the data parameters:
Configurable Options

The following options can be set in DataSim, and sent to the Cogent DataHub. All of the numeric options have a corresponding point in the DataHub that contains the value. Thus, these values can be set from within the Data Browser, by selecting the point name and entering a new value for it.

Waveform Parameters

Amplitude
The height of the wave forms. DataHub Point name: Amplitude.

Offset
An offset from zero of the generated data, and thus the wave form. DataHub Point name: Offset.

Frequency
The frequency of the wave form. DataHub Point name: Frequency.

Data Domain
The Cogent DataHub data domain name for the data points. Any entry in this box changes the connection Status to IDLE. You must press the Reconnect button to reestablish the connection.

Update Frequency
The number of times per second that data changes and is sent to the Cogent DataHub.

Apply Changes
Applies any changes that have been entered for these configurable options. This but-
ton is greyed out until a change has been entered and can be applied.

**Send data change as binary...**

Allows you to send the data changes from DataSim in binary form, rather than as ASCII characters. This can speed data update rates substantially. This feature is only available on x86 machines.

### Data Generation

For each of the following four variables, the check box stops or starts data generation, while the toggle button hides or shows the graph display. The numerical value is shown at the right of the graph.

- **SIN**
  Data generates a sine wave.

- **SQR**
  Data generates a square wave.

- **TRI**
  Data generates a triangular (45 degree) wave pattern.

- **RMP**
  Data generates a ramp wave—steadily increasing, followed by a sudden drop.

### Display Mode

Controls how the data is displayed in the trend display. **Scroll** moves the wave from right to left as the data is generated. Old data scrolls off the left side of the display as new data scrolls in from the right. **Wrap** adds to the wave from left to right, and writes over old data.

### DataPid - a PID loop data simulation program

DataPid is a data simulation program for the Cogent DataHub. It simulates data for a set point, control output, and process variable, and sends these values to the DataHub by a TCP connection.

This program is similar to DataSim, but with specialized data. A PID loop is often used by process control engineers to determine the efficiency of their system. A detailed explanation of PID loops is beyond the scope of this manual, but the data generated by this simulator can be used by anyone.

### Starting Up

The DataPid can be started from the Windows **Start** menu, the command line, or by clicking on the desktop icon.
As soon as DataPid starts, it attempts to connect to a DataHub and begins generating data. To receive the data, the Cogent DataHub should be configured as a tunnelling/mirroring master.

The DataPid can take several command-line arguments at startup, to assist in running more than one instance at a time. Any combination of the following arguments can be supplied in the Target field of the shortcut, or on the command line.

/d data domain name
The name of the DataHub data domain in which to write the data that DataPid generates. The default is DataPid.

/n pid name
The name of the PID loop. For example, if data domain name is DataPid and pid name is PID1 then the data will be created in a hierarchy beneath the point DataPid:PID1. The default is PID1.

/h host name
The name of the computer on which the DataHub is running. This can be an address or a name, for example, 127.0.0.1 or developers.cogentrts.com. The default is localhost.

/p port number
The port number on the target computer on which to connect. The default is 4502.

/i
If specified, the DataPid window will be iconified when it starts.

Example:

DataPid.exe /i /d test /n pid

Connection Parameters

Host
The name or IP address of the host computer. Since DataSim connects via TCP, this can be any computer on the network running a Cogent DataHub acting as a tunnelling/mirroring master.
Service
The port number or service name as entered in the Master service/port entry box of the DataHub.

Status
Displays the attempts to connect, which change to Connected when the connection is made.

You can use the Reconnect button to reconnect and the Pause button to freeze all data generation. Press the More... button to view and change some of the data parameters:

Configurable Options
The following options can be set in DataPid, and sent to the Cogent DataHub. All of the numeric options have a corresponding point in the DataHub that contains the value. Thus, these values can be set from within the Data Browser, by selecting the point name and entering a new value for it.

Setpoint (SP)

- **Manual**
  A number from 0 to 100 to for the set point. An entry here overrides Auto.

- **Auto**
  Changes the set point randomly, every \( n \) seconds, where \( n \) is the number entered.

- **Point**
  Changes the set point according to a point registered in the DataHub.
Controller (MV)

- **Kp**: The proportional control factor, sets the speed of adjustment. The wave form button sets a well-tuned PID loop.
- **Ki**: The integral control factor, reduces error. The wave form button sets a poorly-tuned PID loop.
- **Kd**: The derivative control factor, provides a damping effect. The wave form button sets an oscillating PID loop.

Plant (PV)

- **Kp**: The proportional control factor of the plant.
- **Ki**: The integral control factor of the plant.

Waveform Parameters

- **Amplitude**: Not yet documented.
- **Offset**: Not yet documented.

Data Domain

The Cogent DataHub data domain name for the data points. Any entry in this box changes the connection **Status** to **IDLE**. You must press the **Reconnect** button to reestablish the connection.

Update Frequency

The number of times per second that data changes and is sent to the Cogent DataHub.

Apply Changes

Applies any changes that have been entered for these configurable options. This button is greyed out until a change has been entered and can be applied.

Data Graph

**SP, MV, and PV**

The toggle button hides or shows the graph display. The numerical value is shown at the right of the graph.

Display Mode

Controls how the data is displayed in the trend display. **Scroll** moves the wave from right to left as the data is generated. Old data scrolls off the left side of the display as new data scrolls in from the right. **Wrap** adds to the wave from left to right, and writes over old data.

Remote Config

The Remote Config tool lets you to connect to the DataHub while it is running as a Windows service, or when it is running on a different computer on the network.
When running the DataHub as a Windows service it is commonly not possible to access its configuration dialog. Windows services have restricted ability to interact with the Windows desktop. You can use Remote Config to conveniently configure DataHub while it is running as a Windows service.

All DataHub installations that are configured to allow remote configuration must also enable the DataHub Web Server. Thus, all DataHub V9 licenses now enable the Web Server just for remote configuration, with no additional licenses necessary. The Web Server will not operate for any other purpose without the normal Web Server license, which can be installed separately. In addition, all DataHub V9 and above licenses include a license for a single remote configuration connection.

The remote configuration tool includes the ability to store access credentials to many DataHub installations at once. You can now easily manage many DataHub installations anywhere on your network from a single location.

**Preparation**

Before connecting to a DataHub remotely to configure it, you must first run it as a normal program, not a service, and prepare it in two ways:

**Security**

1. In the **Security** option of the DataHub Properties window, click the **Configure Permissions** button.
This opens the **Edit Permissions** window.

2. In the **UserName** panel, enter a new user name (like RemoteUser), and check the **RemoteConfig** box in **Group Memberships**.
3. Click in the **Password** column to enter a password.

The password for the **RemoteConfig** permissions must be a strong password, defined as:
1. At least 8 characters long, and
2. having mixed character case, and
3. not a capitalized word.
   
   For example, **Password** will not work, but **pAssword**, **PassWord**, **Password123** will. Better yet, use something stronger like **Con#rol%9**.

4. When finished, click **Apply and Close**.

For more information about security, please refer to DataHub **Security** configuration.

### Features

You must configure the DataHub to allow remote configuration for any features that you wish to be able to remotely configure. For example, you may want to remotely configure Tunnel connections, but not allow scripts to be configured except from the local computer. You specify the configurable features as follows:

1. In the DataHub Properties window, select **Remote Config**.
2. Check the boxes of the features you want to be configurable. Checking a box in the **Local** column allows connections only from Remote Config running on the same machine, whereas boxes in the **Remote** column allow connections from Remote Config running on a remote machine. If you uncheck **Local** and check **Remote**, then a user on the local machine will not be able to configure that feature.

![Service and Remote Configuration](image)

Your DataHub is now ready to accept connections from Remote Config. Now you can go to **Configuring a Local DataHub** for configuring a local connection, such as when running the DataHub as a service, or to **Configuring a Remote DataHub** for remote connections.

The Remote Config tool can connect to DataHub Version 8 but cannot configure its settings. The only purpose in connection to DataHub V8 would be to remotely view the Data Browser, Event Log, Connection Viewer or Script Log. To make such a connection, use the button **Enter (Ignore Errors)** when authenticating the connection.

**Configuring a Local DataHub**

When the DataHub is running as a Windows Service, you can use the Remote Config program to interact with it, without having to stop the service. You can reconfigure the DataHub, add new connections or view data and event log messages using the Remote Config program.

Here is how to use the Remote Config program to interact with the DataHub while it is running as a Windows service on the same computer:

1. From the Start menu, go to the Cogent programs and select the RemoteConfig program.

![RemoteConfig](image)

This will open the Remote Config program interface.
Enter the following information

**Name**
A name for this connection, any text string.

**Host**
The URL or machine name of the computer where the DataHub is running.

**HTTP Port**
This port is used to exchange configuration data between the Remote Config program and the DataHub, via the DataHub Web Server. Use port 80 for plain text, unless you have changed the default settings on your Web Server. Entries here may need to be correlated with entries for the **Data Port**. See the table below for more details.

When configuring the DataHub Web Server, you must ensure that the port you configure is not in use. Please see the section called “Configuring the DataHub Web Server” for more information.

You do not need a Web Server license to connect Remote Config to
the DataHub.

**Data Port**

This port is used for passing data between the DataHub and the Remote Config program. The default and recommended setting is to make a WebSocket connection, which will use the same port as you choose for HTTP. You must use a WebSocket connection for the data port if your DataHub license does not enable the Tunnel/Mirror feature. Here are some of the possible HTTP and data port combinations.

<table>
<thead>
<tr>
<th>HTTP Type - with - Data Type</th>
<th>HTTP Port</th>
<th>Data Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain text - with - Plain text</td>
<td>configure 80, no SSL</td>
<td>4502, no SSL, no WebSocket</td>
</tr>
<tr>
<td>Plain text - with - WebSocket</td>
<td>configure 80, no SSL</td>
<td>80, use WebSocket</td>
</tr>
<tr>
<td>SSL - with - SSL, no WebSocket</td>
<td>configure 4503, use SSL</td>
<td>4503, use SSL, no WebSocket</td>
</tr>
<tr>
<td>SSL - with - SSL and WebSocket</td>
<td>configure 443, use SSL</td>
<td>443, use SSL and WebSocket</td>
</tr>
</tbody>
</table>

**User Name**

A name for a DataHub user that has administrative permissions (see Preparation for details).

**Password**

The password for that user. Use the **Save Password** option to avoid re-entering it each time.

**Accept untrusted certificates**

Checking this box allows the Remote Config program to attempt a connection with a DataHub that may be using certificates for secure connections, such as the OPC UA server feature.

2. Press the **Enter** button to make the connection. You should now see a remote configuration version of the DataHub Properties window:
The panel on the left lists all of the DataHub connections you have configured. The bottom panel displays the application log for this connection to the DataHub. The main panel displays the Remote Config copy of the DataHub Properties window for the DataHub you are connected to.

Pressing the **Enter (Ignore Errors)** button instead of the **Enter** button when you log in (as described above) will allow you to open the Data Browser, and possibly the Event Log, Connections, and Script Log. But you will not be able to see or change any configuration.

**Differences between Remote Config and the native DataHub Properties windows**

The Remote Config program is very similar to the native DataHub Properties window, with the following exceptions;

- **Reduced feature set** Not all of the DataHub features are replicated in Remote Config. Some are absent for security reasons, and others may be added later.

- **Database table creation** You cannot create a database table through Remote Config
as you can with **Database** in the native Properties window. Instead, create a table using the management tools that come with the target database, then assign points to the columns in Remote Config.

- **Apply button** The **Apply** button takes the place of both **OK** and **Apply** from the native Properties window. It applies changes only for the configuration panel that is currently visible.

- **Refresh button** The **Refresh** button takes the place of the **Cancel** button from the native Properties window, and does two things. It reloads the current configuration, and discards any changes that have not yet been applied.

- **About** is now at the bottom of the list of DataHub options.

- **Data Browser** There is a 10 - 15 second delay for data domains to appear. Also, there is no **Drag and Drop** button.

A point filter helps you find point(s) you are interested in within a selected domain.

**View Connections** The statistics in this window run on a 5-second cycle, instead of a 1-second cycle like the native window.

**Script Log** A panel on the left displays timestamps that get created whenever new output is printed to the screen. Checking the **Submit with Ctrl-Enter** box allows you to make multi-line entries.

**Connection status** When viewing connections in the Remote Config program (for example tunnel connections), the status of a connection only gets updated when you click the **Refresh** button.

### Configuring a Remote DataHub

You can configure a DataHub from a remote computer with the Remote Config app. A special program, the Skkynet Web Application Manager, that comes with the DataHub helps you securely manage the process. Once you have prepared a remote DataHub, you
can download the Web Application Manager from it, and web launch the Remote Config app, as follows:

1. Ensure that the DataHub instance you want to access is running, and that its Web Server feature is enabled.

2. Open a web browser and type in the IP address or network name for the DataHub computer. If the Web Server feature on that DataHub is configured to use a port other than the default port 80, include that port number in the network address. For example, if the remote DataHub is at 192.168.3.12 and its Web Server is configured to use port 8080, then the network address would be 192.168.3.12:8080.
   This opens the Web Launch Page.

3. Click the Download Web Application Manager button and follow the instructions to download and install the Skkynet Web Application Manager. Now you are ready to launch an application.

   This installation is signed with a valid code signing certificate from Skkynet. If Windows cannot validate the certificate, do not install the software.
Launching the Remote Config app

Having installed the Web Application Manager, you can now follow these steps:

1. From the Web Launch Page, click on the **Launch Remote Config** button. A dialog will ask if you want to switch to the Web Application Manager.
2. Click **Accept**. The Web Application Manager will open, populated with the parameters required for you to connect to the remote DataHub system and download the Remote Config app.
3. Click the green **Run** button.
   The files needed to run the Remote Config app will be downloaded, and a dialog window will display information about the program files, with details of the security certificate used to sign them.

   Please see SSL Certificates for more information about SSL certificates in...
Other Windows and Programs

4. Check the signing certificate and affected files to determine whether or not to accept them, based on the following criteria:

   • If the **Signing Certificate** region of this dialog is not green, then the **Status** will be **Invalid**. That indicates that the certificate cannot be verified. Do not click **Accept** unless you know why you are seeing a rejected certificate.

   • If the **Signing Certificate** is from an organization other than Skkynet Cloud Systems Inc. then it indicates that another company's software has been installed on the server and is being delivered as part of the application. If you do not believe this to be legitimate, do not click **Accept**.

   • **If you click Accept** it means that you trust the app source files. They will be saved on your computer and the app will run. Your choice will be remembered, and the next time you web launch this app you will not have to accept the files again.

   ![Signed DLL Certificate]

   If, for some reason you want to revoke your acceptance of the downloaded files, you can select the profile in the Web Applications Manager and click the **Clear Authorization** button.

   • **If you click Reject** the files required to run the app will be deleted. You will also see a list of files you just rejected and be given the chance to change your mind. To revoke your choice:

     1. Click the **Reset DLL Authorization** button.
     2. Click **OK** to close the dialog window.
     3. Download the files again by clicking the green **Run** button in the Web Application Manager.

5. Proceed with configuring the remote DataHub, as explained in **Configuring a Local DataHub**.
Create a Shortcut

Here are two ways to make a shortcut to quickly start Remote Config.

1. While Remote Config is running, you can pin the program to your task bar, creating a shortcut.
2. From the Web Launch Page you can drag the direct link: nnn.nnn.nnn.nnn.config, onto your desktop.

With either of these shortcuts you can start Remote Config with one click; no need to visit the Web Launch Page.

Service Manager

The Service Manager is a program that provides access to Cogent software that can be installed to run as a Windows service. With this program you can select and configure how the service runs, change its status, and open the Properties window of the Cogent DataHub.

Please see Remote Config for the most convenient way to configure the DataHub while it is running as a service.

If you are upgrading to a newer major version of the DataHub, please refer to Running DataHub as a service in Windows 10 and Server 2016 in Known Issues on the Cogent DataHub website for more information.

Starting the Service Manager

The Service Manager can be started from the Cogent DataHub program group in the Cogent entry of the Windows Start menu.
Once started, you can choose the service you need to configure from the **Select Service** dropdown list at the top. Then you can configure, install, and check the status of the service, as well as view the DataHub Properties window. The scrolling list at the bottom maintains a record of activities.

**Service Configuration**
Startup type:
Choose between Automatic, Manual, or Disabled to specify how you want the service to start when Windows starts.

Configuration folder:
Allows you to specify a folder in which to put the configuration files for the Cogent DataHub. Typically this does not need to be changed. Please refer to Configuration Files for more information.

The Set Global Configuration Folder button lets you specify the configuration file through the registry. This will override the default configuration folder, but will not override a configuration folder supplied on the DataHub command line with the -H option. You can use environment variables to make this folder dependent on the system and user configuration. The Sample field gives an example of what the folder path will likely be, but it is not necessarily accurate. If the DataHub is run as a specified user, and the path is dependent on the user, then the Sample will display a path based on the current user, not the executing user, and will therefore show an inaccurate path. You can find the true configuration path in the DataHub About panel, by clicking the About button in the DataHub Properties window.

Run as specified user
If you run the DataHub as the local SYSTEM (LocalSystem) user, then you will be able to access the Properties window and make changes to the DataHub while it is running as a service. If you run as any other user, then you will need to access the Properties window using the Remote Config option.

Automatically restart on failure
Have the service restart should it fail or be stopped for some reason.

Install/Update Service
When the above configuration is complete, press this button to install the DataHub as a service. This will also start the DataHub service, though on some systems the service may need to be started manually if it doesn't start with the install operation. If the service is running and you make changes to the configuration, pressing this button will cycle through a service shutdown and restart to apply your changes.

Service Status

![Service Status](attachment:image)

Installed as a service:
Indicates whether the selected program is installed as a service or not (True or False). The Uninstall button allows you to uninstall the DataHub as a service.

Service state:
Indicates the run status of the service (Stopped, StartPending, Running, etc.) . The Start and Stop buttons allow you to start or stop the service.
Show Properties Dialog

This option appears differently for different versions of the Windows operating system:

Windows XP and Windows Server 2003

A Show Properties Dialog button allows you to open the Cogent DataHub Properties window.

The DataHub Properties window is only visible on the primary console of the computer running the Cogent DataHub. If you are currently logged in via a remote desktop session, you will see a pop-up dialog indicating that you must be connected to the computer's primary console. You can do this by using the /admin or /console options on the Microsoft Remote Desktop client.

If you are already connected to the primary console, the DataHub Properties window and system tray icon will be displayed when you press Show Properties Dialog button.


The Switch to Service Console button displays the Windows Service Console, which is where the DataHub Properties window appears when running as a service. This allows you to view data and make changes to the DataHub configuration while it is running as a service.

The Service Console is a special display console provided by Microsoft Windows, also known as the "session 0 console". This was introduced in Windows Vista as a security mechanism to limit access to the user interface of high-permission processes. When you switch to the Service Console, your desktop will be hidden and the screen background will change color to indicate the special status of the Service Console.
While the Service Console is open, your other applications will continue to run normally, but you will not be able to see or interact with them. A dialog box will be visible while you are viewing the Service Console that will allow you to switch back to your regular desktop at any time. If you have other system services running on your computer that also have user interface windows, those services will also be available to you while viewing the Service Console.

When you have finished viewing or editing the DataHub properties, before returning to the normal user console, you should click the **Apply** and **OK** buttons to close down the DataHub Properties window. Also be sure to close any other DataHub windows. This prevents the Windows Interactive Service Detection program from displaying pop-up messages when you return to the normal user console.

Once all the DataHub windows are closed, you can return to the normal user console by pressing the **Return Now** button in the Interactive Service Detection window.

If you forget to close any of the DataHub windows while working in the Service Console, Windows will begin popping up messages in the user console telling you there is a program requiring attention in the Service Console.
You can stop these messages and close down the Service Console by clicking on the **Disable Service Console** button in the DataHub Service Manager. Clicking this button will not stop the DataHub service.

When you close the Service Manager it will automatically disable the service console. This will stop Windows from periodically displaying the Interactive Services Detection dialog on your system.
Tunnelling OPC DA

Introduction

When talking about data networks, *tunnelling* means to encapsulate one protocol inside another, to allow it to be sent more easily and/or securely across the network. The Cogent DataHub tunnels OPC DA to avoid the problems of DCOM configuration commonly encountered when using OPC DA.

Click here to watch a video.

Using the Cogent DataHub for tunnelling OPC DA means:

- Connect equally easily across a LAN or WAN.
- No DCOM—no timeouts or configuration problems.
- Complete and secure data access.
- Simple set up.

Setting up the Cogent DataHub for tunnelling OPC DA is a simple, 3-step configuration that can be done in a few minutes. All you need to do is:

1. Configure the DataHub on the OPC DA server machine.
2. Configure the DataHub on the OPC DA client machine.
3. Start the OPC DA client.

Why use tunnelling for OPC DA?

Most people who attempt to network OPC DA servers and clients experience a number of problems. Networking is just not OPC DA's strength. OPC DA was originally based on COM (component object model), which runs on a single computer.

As long as the OPC DA server and client are on one computer, setting up the connection between them is easy. Difficulties arise, however, when an OPC DA server and client are on different computers, and need to be networked.
To communicate over a network, OPC DA uses DCOM (distributed COM), which many system engineers find to be inadequate for their needs. DCOM can time out for up to 5 minutes at a time, and there are problems related to running computers in different security domains. As the number of networked OPC servers and clients increases, the difficulties with DCOM increase exponentially. It is very difficult over a LAN, and most people don't even attempt it over a WAN.

The Cogent DataHub provides a COM (OPC DA) interface for the OPC DA client and server, and uses DHTP over TCP across the network. This is known as tunnelling. Each connected OPC DA server or client sees the other as a local OPC connection. They are unaware that their messages are being converted to TCP along the way. Tunnelling works equally well across a LAN or WAN, and it is even possible to tunnel through firewalls. See Typical Scenarios for more information.

**Configuring the DataHub for the server**

Configuring tunnelling on the server side comprises two tasks: making the OPC server connection and configuring the tunnel/mirror connection.

**Configure the DataHub the OPC DA connection**

The DataHub on the OPC DA server machine will act as an OPC client. You should configure it as follows:

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **OPC DA**.
3. Check the **Act as an OPC Client** box. Since the DataHub can be a client to more than one OPC server, you need to specify which OPC DA server you are going to connect to. To add a server, click the **Add** button and fill in the fields in the **Define OPC Server** Window:

4. Type in or select the necessary information as appropriate.
   a. The first four fields define the OPC server:
• **Connection Name** Type a name to identify this connection. There should be no spaces in the name. It doesn't matter what name is chosen, but it should be unique to other connection names.

• **Computer Name** Type in or select from the drop-down list the name or IP address of the computer running the OPC server you want to connect to.

• **OPC Server Name** Select the name of the OPC server that you are connecting to from the list of available servers.

• **Data Domain Name** Type the name of the DataHub data domain in which the data points will appear.

b. You can specify how the data is to be transferred.

• **Maximum update rate (milliseconds)** Enter the maximum rate you wish the data to be updated. This is useful for slowing down the rate of incoming data. The default is 0, which causes values to be updated as soon as possible. This value is also the polling time used by asynchronous and synchronous reads (see below).

• **Read Method** Choose how to read data from the OPC server:

  • **Asynchronous Advise** The OPC server sends a configured point's data to the DataHub immediately whenever the point changes value. This is the most efficient option, and has the least latency.

  • **Asynchronous Read** The DataHub polls the OPC server for all configured points on a timed interval (set by the **Maximum update rate**). This option is less efficient than Asynchronous Advise, and has higher latency.

  • **Synchronous Cache Read** The DataHub polls the OPC server for all configured points on a timed interval (set by the **Maximum update rate**), and this thread waits for a reply. This option is less efficient than Asynchronous Advise or Read, and has higher latency than either of them.

  • **Synchronous Device Read** The DataHub polls the PLC or other hardware device connected to the OPC server for all configured points on a timed interval (set by the **Maximum update rate**), and this thread waits for a reply. This
is the least efficient of all of these options, and has the highest latency.

- **Write Method** Choose how to write data to the OPC server:
  - **Asynchronous Write** provides higher performance. The Cogent DataHub writes changes in point values to the OPC server without waiting for a response.
  - **Synchronous Write** elicits a quicker response from the OPC server, but results in lower overall performance. The Cogent DataHub writes changes in point values to the OPC server without waiting for a response. This option is useful if the OPC server doesn't support asynchronous writes at all, or if it can't handle a large number of them.

Depending on the OPC server you are configuring, you might have an option to use OPC DA 2.0 or 3.0. Please refer to the Data Transfer explanation in the OPC section of the Properties Window chapter for more information.

c. There are several optional entries:

- **Treat OPC item properties as DataHub points** lets you register and use non-standard OPC item properties as points in the DataHub. Generally you won't need this unless you plan to use the DataHub to distribute changes to values of the non-standard properties on your OPC items.

  The Cogent DataHub will monitor these properties only if the OPC server exposes them as OPC items. If the properties do not show up when using this check-box, this means that the server does not expose the non-standard properties as items.

  Some OPC DA servers are slow to register their OPC items and properties. Using this option with one of these servers can significantly slow the start-up time of the DataHub

- **Read only: Mark all items as Read-Only** lets you specify that the OPC server be read-only, regardless of how individual items are specified. Items in the DataHub that originate from such an OPC server will be read-only to all DataHub clients.

- **Replace item time stamps with local clock time** allows you to set the time-
stamps for the items from this server to local clock time.

- **Force connection to use OPC DA 3.0** lets you choose the DA 3.0 write methods from the **Write Method** drop-down box. It will also instruct the Cogent DataHub to attempt to browse the server using DA 3.0 browsing. This setting will override any automatic information that the Cogent DataHub may determine about the server based on the server’s registry entries.

- **Never use OPC DA 3.0** removes the DA 3.0 write methods from the **Write Method** drop-down box, and will instruct the Cogent DataHub to only use DA 2.0 browsing. This setting will override any automatic information that the Cogent DataHub may determine about the server based on the server’s registry entries.

For more information about OPC DA 2.0 and 3.0, please refer to the **Data Transfer** explanation in the OPC section of the Properties Window chapter.

- **Set failed incoming values to zero** The OPC DA spec requires an OPC DA server to send an *EMPTY* (zero) value whenever it sends a failure code in response to an item change or a read request. Some OPC servers, however, send a valid value with the failure code under certain circumstances. To ignore any such value from the OPC server and assume *EMPTY*, keep this box checked (the default). If instead you want to use the value supplied by your OPC server, uncheck this box.

  ![Warning]

  Unchecking this box will make the Cogent DataHub’s behavior non-compliant with the OPC specification.

- **Never use OPC DA 2.0 BROWSE_TO function** disallows the **BROWSE_TO** function when communicating with OPC DA 2 servers. Sometimes an OPC server will have problems with this function that prevent the Cogent DataHub from connecting to it. Checking this box might allow the connection to be established in those cases.

- **Never attach to an in-process COM server** Most vendors include both an in-process and out-of-process COM server with their OPC DA server installation. If both options are available, the DataHub connects to the in-process server, as it is generally the better choice. This option forces the DataHub to consider only out-of-process servers.

  Why is this useful? An in-process server is implemented as a DLL that is loaded into the client's address space. This makes the client very dependent on the good implementation of the server. If there is a crash in an in-process server, the client also crashes. An out-of-process server is implemented as a separate executable. The client communicates with an out-of-process server using the inter-process communication mechanisms in DCOM. In theory an in-process server will be faster than an out-of-process server, but sometimes the in-process server is less robust than the out-of-process server and leads to instability or malfunction in the client.
• **Allow VT_EMPTY canonical type for OPC DA2** The VT_EMPTY canonical type may be incompatible for a particular combination of OPC server and client. For example, some clients or servers that were built before 64-bit integers were common may fail when presented with a 64-bit number. These options (DA2 and DA3) allow you to enable or disable the VT_EMPTY canonical type, either for trouble-shooting or as a permanent part of your configuration.

• **Allow VT_EMPTY canonical type for OPC DA3** See above.

• **Wait for server running state** Every OPC DA server takes a little time to initialize before it will allow client connections. This option lets the user specify the time to wait for the OPC server to initialize. The wait time is a maximum; if a server initializes before this time, the DataHub will connect right away. If the server doesn't initialize within this time, the DataHub will report this in the Event Log, and then try to connect anyway.

• **Pause before reading data** specifies a time for the DataHub to pause before reading the OPC server's data set. Some OPC DA servers report that they are running, but have not yet received the full data set from the process. If the DataHub attempts to connect right away, it might get a partial data set. The pause is fixed; it will always last for the full time specified.

The two above times are added together. The DataHub will wait until the server is initialized (or until the specified "wait" period is complete) and then pause for the specified "pause" time, before trying to read data from the server. For example, with the defaults of 5000 and 1000, at least 1 second and at most 6 seconds will elapse before the DataHub tries to read the data set.

d. Finally, you can specify how the OPC items get selected. You can select them manually or load all of them.

![Item Selection](image)

**Manually Select Items**

Check the **Manually Select Items** box and press the **Configure Items** button to open the OPC Item Selection window, where you can specify exactly which
You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. Follow these guidelines for making selections:

• To select a server item from the right-hand pane, click its check-box.

• To highlight a list of consecutive server items, click the first item, hold down the \texttt{Shift} key, and then click the last item. To highlight separate server items, hold down the \texttt{Ctrl} key as you click each item. To select a group of highlighted items, use the \texttt{Spacebar}.

These may not function as described for Windows NT or Windows 2000 operating systems.

• Selecting a server item does not automatically add any of its child items. Each child item must be added separately. To view child items, click the + sign in front of the item. If an item has one or more children that have been selected, the item name(s) will appear in bold.

• To delete selected items from the right-hand pane, highlight them and press the \texttt{Remove Selected} button. Use the \texttt{Shift} and \texttt{Ctrl} keys as above to highlight groups of selected items.

You may also configure dynamic items on the server. As you type in the \texttt{Server Item ID}, the system will fill in an identical \texttt{DataHub Point Name} for you (which you can change at any time). Press the \texttt{Enter} key or the \texttt{Apply} button to create the item. Checking the \texttt{Copy names from selection} box will fill in...
the entry with the name you select from the **Selected Items** list (above). The **Recognize branch delimiter in point name** option lets you select and apply a point delimiter for your dynamic items.

### Load All Items on Server

In addition to manually loading items, you have the option in the Define OPC Server dialog to register all points, or filter for groups of points, from the OPC server.

![Configuring Items](image)

In the **Server specific item filters** you have the option create filters to select partial data sets. If you don't enter anything here, the DataHub will query the OPC server for all of its items and register them. The filters are all applied on a logical 'OR' basis, i.e. if a point satisfies the condition of any filter, it gets registered with the DataHub.

- Click the **Add...** button to add a filter. The **Edit a filter string** window will appear:

![Edit a filter string](image)

Enter a string or a pattern to match one or more item names in the OPC server. Each server has its own syntax for pattern matching, so you may have to experiment a little to get exactly the points you need. Commonly, the symbol * matches any number of characters, while the symbol ? often matches a single character. In that case, an entry of ?a* would bring in all items with a as the second letter in their names.

- Click the **Edit...** button to open the **Edit a filter string** window and edit an existing filter. You can do the same thing by double-clicking a filter string in the list.

- Click the **Remove** button to remove a selected filter from the list.

5. Click the **Apply** button in the Properties Window. The DataHub should begin to act as a client to the OPC server. Messages will appear in the **Status** column indicating the status of the connection:

**Configuring** After you click the **OK** button in the Define OPC Server dialog until you
click the **Apply** button in the Properties window.

**Server Lookup** The DataHub is looking for the OPC server.

**Server Attach** The DataHub has found the OPC server and is connecting. It may be waiting for the server running state, as explained previously.

**Pause NNNN ms** The DataHub is paused before reading data, as explained previously.

**Running** The DataHub is connected to the OPC server and exchanging data.

**Disconnected** The DataHub has disconnected from the OPC server.

You can verify the connection using the Data Browser or the Connection Viewer. You can change server settings at any time. The Cogent DataHub will reconnect and apply the changes when you click the **Apply** button in the Properties Window.

**Configure the DataHub as tunnelling master**

The tunnelling master DataHub receives the initial request from a tunnelling slave to establish the tunnelling connection, initially or after a network break. For this reason we suggest that for any two tunnelling DataHubs, the master be on the OPC server machine. Once the connection is established the two DataHubs are indistinguishable from each other; both DataHubs will send and receive data changes.

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **Tunnel/Mirror**.
3. In the **Tunnelling Master** section, you can configure plain-text or secure tunnelling. Ensure that at least one of these is checked. If you want to change any of the other defaults, please refer to the section called “Tunnel/Mirror“ for more information.

   ![Tunnel/Mirror](image)

   To optimize throughput, un-check the **Try to send data even if it is known to be superseded** option. This will allow the DataHub to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

4. To support incoming WebSocket connections from DataHub tunnelling clients, you will need to configure the tunnelling master DataHub's **Web Server**. For WebSocket
connections, we recommend using SSL, on port 443.

5. Click **OK** to close the Properties window.

The server machine side of the tunnelling connection is now ready, and you can move to the client machine.

**Configuring the DataHub for the client**

Configuring tunnelling on the client side also comprises two tasks: configuring the tunnel/mirror connection and making the OPC client connection.

Now you need to set up the Cogent DataHub on this machine to tunnel across to the DataHub on the OPC DA server machine.

**Configure the Cogent DataHub as tunnelling slave**

The tunnelling slave DataHub behaves exactly like the tunnelling master DataHub except that the slave establishes the tunnelling connection initially, and reestablishes it after a network break. For this reason we recommend that the DataHub on the client side act as the tunnelling slave, while the DataHub on the server side act as the tunnelling master.

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **Tunnel/Mirror**.

![Tunnel/Mirror Configuration](image)

3. Check the box **Act as a tunnelling/mirror slave to these masters**.
4. Click the **Add Master...** button to assign a master to this slave. The **Tunnel/Mirror Master Configuration** window will open:
5. Type in the following information:

- **Primary Host** the name or IP address of the computer running the tunnelling master DataHub.

- **Port** the port number or service name for this host. You should use default port number (4502) unless you have changed the entry in the master DataHub.

- **Secondary Host** gives you the option to have an alternate host and service/port number. On startup or after a network break, the DataHub will search first for the primary host, then for the secondary host, alternating between primary and secondary until a connection is made. If no secondary host is specified, the connection will be attempted on the primary host only.

This feature is not recommended for implementing redundancy because it only checks for a TCP disconnect. The DataHub **Redundancy** feature, on the other hand, provides full-time TCP connections to both data sources, for instantaneous switchover when one source fails for any reason. There is no need to start up the OPC DA server and wait for it to configure its data set. You can also specify a preferred source, and automatically switch back to that data source whenever it becomes available. By contrast, the primary and secondary host in the tunnel can act as a primitive form of redundancy, but will only switch on a connection failure at the TCP level, which is only one sort of failure that a real redundancy pair must consider.

- **Local data domain** The data domain in which you plan to receive data.

- **Remote data domain** the master DataHub data domain from which you plan to receive data. Point names will be mapped from the remote data domain (on the master DataHub) into the local data domain (on this DataHub), and vice versa.

Unless you have a good reason for making these different, we recommend using the same data domain name on both DataHubs for the sake of simplicity.
• **Remote user name** The user name for TCP security, established on the tunnelling master, using the DataHub Security option in the Properties window.

• **Remote password** The password for TCP security, established on the tunnelling master, using the DataHub Security option in the Properties window.

• **Secure (SSL)** lets you establish a secure connection using SSL tunnelling as long as the tunnelling master DataHub you are attempting to connect to has been configured for secure connections. The additional options allow for a connection to be made even if the security certificate is invalid, or the host name does not match. We don't recommend using these options unless absolutely necessary. For more about SSL, please refer to the section called “SSL and Firewalls”.

• **WebSocket** lets you connect via WebSocket. This option is applied for both primary and secondary hosts, and allows you to enter a **Proxy address**, and a **Proxy port** number, **username**, and **password** as needed. When tunnelling through a proxy, HTTP uses normal HTTP proxy, and HTTPS uses HTTP CONNECT proxy. You can select the **Always use HTTP CONNECT** to use it for HTTP as well as HTTPS.

> The WebSocket protocol requires a web server to act as an intermediary. So, for this option you will need to use the DataHub Web Server on the tunnelling master DataHub (as explained here).

There is a DataHub running on a Skkynet cloud server that you can connect to for testing. Here are the parameters you will need to enter for it:

• **Primary Host** demo.skkynet.com

• **Port** Will be set automatically by the system, 80 for **WebSocket** and 443 for **Secure (SSL)**.

• **Local data domain** cloud

• **Remote data domain** DataPid

• **Remote user name** demo/guest

• **Remote password** guest

• **WebSocket** Must be selected.

• **Secure (SSL)** Optional.

6. You now have several options for the mirrored connection.
a. **Data Flow Direction** lets you determine which way the data flows. The default is bi-directional data flow between slave and master, but you can effectively set up a read-only or write-only connection by choosing that respective option.

To optimize throughput, check the **Read-only Receive data from the Master, but do not send** option. Only do this if you actually want a read-only connection. If you do not require read-write access, a read-only tunnel will be faster.

b. **When the connection is initiated** determines how the values from the points are assigned when the slave first connects to the master. There three possibilities: the slave gets all values from the master, the slave sends all its values to the master, or the master and slave synchronize their data sets, point by point, according to the most recent value of each point (the default).

c. **When the connection is lost** determines where to display the data quality as "Not Connected"—on the master, on the slave, or neither.

If you have configured **When the connection is initiated** as **Synchronize based on time stamp** (see above), then this option must be set to **Do not modify the data quality here or on the Master** to get correct data synchronization.

d. **Connection Properties** gives you these options

- **Replace incoming timestamp...** lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.

- **Transmit point changes in binary** gives users of x86 CPUs a way to speed up
the data transfer rate. Selecting this option can improve maximum throughput by up to 50%.

For more information, please refer to the section called “Binary Mode Tunnel/Mirror (TCP) Connections”.

- **Target is an Embedded Toolkit server** allows this slave to connect to an Embedded Toolkit server rather than to another DataHub.
- **Heartbeat** sends a heartbeat message to the master every number of milliseconds specified here, to verify that the connection is up.
- **Timeout** specifies the timeout period for the heartbeat. If the slave DataHub doesn’t receive a response from the master within this timeout, it drops the connection. You must set the timeout time at least twice the heartbeat time.

To optimize this setting, please refer to the section called “Tunnel/Mirror (TCP) Heartbeat and Timeout”.

- **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

7. Click **OK** to close the **Tunnel/Mirror Master** window. The fields in the **Tunnelling Slave** table of the Properties Window should now be filled in.

8. Click the **Apply** button in the Properties Window. If the master DataHub is running, this DataHub should establish the tunnelling connection, and the **Status** should display **Connected**. You can view the data with the **Data Browser**, or view the connection with the **Connection Viewer**.

**Configure the Cogent DataHub for OPC DA**

Finally, we suggest that you ensure that the Cogent DataHub on the OPC DA client machine is configured to act as an OPC server. Every Cogent DataHub comes preconfigured that way, but it doesn’t hurt to check.

1. Right click on the DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **OPC DA**.

3. Ensure that the **Act as an OPC Server** box is checked.
If your OPC client requires that you hand-enter the OPC server name, use either `Cogent.CogentDataHub` or `Cogent.CogentDataHub.1`.

4. For information on any of the other options, please refer to the [OPC DA Server](#) section in Properties.

5. Click **Apply** button at the bottom of the Properties window to apply the change. You can view connections with the **Connection Viewer**.

Now you can start your OPC DA client, connect to the DataHub, and access your data.

### Testing the connection

You can test your tunnelling connection like this:

1. Ensure that you have correctly configured the DataHub on the [OPC server machine](#) and the [OPC client machine](#).

2. Start the DataHub on the OPC server machine if it is not running already. It should start up the OPC server on that machine.

3. Start the OPC client on the OPC client machine. It should start the DataHub, and once the connection has been established the data from the OPC server should be visible in the OPC client.

4. You can view connections with the **Connection Viewer**.

If you don't see data in your OPC client, double-check the following:

- The DataHub configuration on both machines.
- The functioning of the OPC server and client.
- The physical network connection.

### Combinations

Here are some ways to create multiple tunnels, and/or integrate other DataHub features.

#### Multiple connections

- You can aggregate data from multiple remote servers into a single DataHub, and bring all the data into one client.
This scenario involves configuring three OPC server machines and one OPC client machine for tunnelling. Aggregation takes place on the OPC client machine.

• You can also use tunnelling to bring data from a single server to many remote clients.

This scenario involves setting up one OPC server machine and three OPC client machines for tunnelling. Aggregation takes place on the OPC server machine.

• You can link multiple DataHubs together to form a daisy chain.
The middle DataHub in this scenario is configured as both a tunnelling slave and a tunnelling master.

- See Typical Scenarios for scenarios that let you tunnel through firewalls and DMZs, keeping all outbound firewall ports closed. That section also covers redundant tunnelling connections.

**Bridging**

Bridging means linking data from one server to another server, usually on a single machine. However, you can also bridge two OPC servers over a network using a tunnelling connection:

This scenario involves setting up the DataHubs on both machines to act as OPC clients to the respective OPC servers. The DataHubs then interface with each other over a tunnelling connection. Configure the DataHub on the machine with the most uptime to be the tunnelling master and the other DataHub to be the tunnelling slave. We recommend that all bridges be configured on just one of the DataHubs.

**Partial data sets**

You often don't need to tunnel all of the data from an OPC server across the network. It is faster and takes less bandwidth to transfer only the data you need. The Cogent DataHub allows you to do this by setting up a separate data domain for the tunneled data. In fact, you can aggregate parts of data sets from several servers into a single data domain, and then tunnel that combined data set across the network.
Putting data from one or more servers into a separate data domain is done through bridging. When you configure bridges, just make sure to create target points in a new, separate data domain.
Tunnelling Scenarios

Tunnelling - Firewalls Open

Tunnel Scenario 1 - Firewalls open, read-only

**Primary Use** Connect OPC DA or A&E on a secure network, avoid DCOM (monitoring only).

**Configuration**

- **OPC Client:** Configure this DataHub as an OPC client to the OPC server.
- **Tunnel Master:** Configure this DataHub as a Tunnel master, to receive connections.
- **OPC Server:** Configure this DataHub as an OPC server for the OPC client.
- **Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates the connection.

**Data Flow Direction:**
- Read-only
- **When Connection Initiated:** Get all values from Master
- **When Connection Lost:** Mark data here “not connected”
**Tunnel Scenario 2 - Firewalls open, read/write**

**Primary Use** Connect OPC DA or A&E on a secure network, avoid DCOM (monitoring and supervisory control).

**Configuration**

**OPC Client**: Configure this DataHub as an OPC client to the OPC server.

**Tunnel Master**: Configure this DataHub as a Tunnel master, to receive connections.

**OPC Server**: Configure this DataHub as an OPC server for the OPC client.

**Tunnel Slave**: Configure this DataHub as a Tunnel slave, so that it initiates the connection.

**Data Flow Direction**: Read-Write

**When Connection Initiated**: Get all values from Master

**When Connection Lost**: Mark data here “not connected”
Tunnelling - One Firewall Closed

Tunnel Scenario 3 - Data source firewall ports closed, read-only

**Primary Use** Securely access OPC UA, DA, or A&E data from outside the control network, without VPNs (monitoring only).

**Configuration**

**OPC Client:** Configure this DataHub as an OPC client to the OPC server.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates the connection.

**Data Flow Direction:**
Write-only

**When Connection Initiated:**
Override Master’s values with mine

**When Connection Lost:**
Mark Master’s data “not connected”

**Configuration**

**OPC Server:** Configure this DataHub as an OPC server for the OPC client.

**Tunnel Master:** Configure this DataHub as a Tunnel master, to receive connections.
Tunnel Scenario 4 - Data source firewall ports closed, read/write

**Primary Use** Securely access and write back to OPC UA, DA, or A&E data from outside the control network, without VPNs (monitoring and supervisory control).

**Configuration**

**OPC Client:** Configure this DataHub as an OPC client to the OPC server.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates the connection.

**Data Flow Direction:**
Read/Write

**When Connection Initiated:**
Override Master's values with mine

**When Connection Lost:**
Mark Master's data “not connected”

**Configuration**

**OPC Server:** Configure this DataHub as an OPC server for the OPC client.

**Tunnel Master:** Configure this DataHub as a Tunnel master, to receive connections.
Tunnelling Scenarios

Tunnelling - Both Firewalls Closed, with DMZ

Tunnel Scenario 5 - Data source and user firewall ports closed, using DMZ, read-only

**Primary Use** Securely transmit OPC UA, DA, or A&E data between secure networks, without VPNs, privately hosted (monitoring only).

![Diagram of Tunnelling Scenario 5](image)

**Configuration**

**OPC Client:** Configure this DataHub as an OPC client to the OPC server.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates connections.

**Data Flow Direction:**
- Write-only

**When Connection Initiated:** Override Master’s values with mine

**When Connection Lost:** Mark Master’s data “not connected”

**DMZ**

**Tunnel Master:** Configure this DataHub as a Tunnel master, to receive connections.

**Configuration**

**OPC Server:** Configure this DataHub as an OPC server for the OPC client.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates connections.

**Data Flow Direction:**
- Read-only

**When Connection Initiated:** Get all values from Master

**When Connection Lost:** Mark data here “not connected”
Tunnel Scenario 6 - Data source and user firewall ports closed, using DMZ, read/write

**Primary Use** Securely transmit OPC UA, DA, or A&E data between secure networks, without VPNs, privately hosted (monitoring and supervisory control).

**Configuration**

- **OPC Client**: Configure this DataHub as an OPC client to the OPC server.
- **Tunnel Slave**: Configure this DataHub as a Tunnel slave, so that it initiates connections.
- **Data Flow Direction**: Read/Write
- **When Connection Initiated**: Override Master's values with mine
- **When Connection Lost**: Mark Master's data “not connected”

**DMZ**

- **Tunnel Master**: Configure this DataHub as a Tunnel master, to receive connections.
- **Data Flow Direction**: Read/Write
- **When Connection Initiated**: Get all values from Master
- **When Connection Lost**: Mark data here “not connected”

**Configuration**

- **OPC Server**: Configure this DataHub as an OPC server for the OPC client.
- **Tunnel Slave**: Configure this DataHub as a Tunnel slave, so that it initiates connections.
Tunnelling - Both Firewalls Closed, with SkkyHub

Tunnel Scenario 7 - Data source and user firewall ports closed, using SkkyHub, read-only

**Primary Use** Securely transmit OPC UA, DA, or A&E data between secure networks, without VPNs, cloud hosted (monitoring only).

**Configurations**

**OPC Client:** Configure this DataHub as an OPC client to the OPC server.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates connections.

**SkkyHub**

**Tunnel Master:** SkkyHub always acts as a Tunnel master, receiving incoming connections.

**Configuration**

**OPC Server:** Configure this DataHub as an OPC server for the OPC client.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates connections.

**Data Flow Direction:**

*Write-only*

*When Connection Initiated:* Override Master’s values with mine

*When Connection Lost:* Mark Master’s data “not connected”
Tunnel Scenario 8 - Data source and user firewall ports closed, using SkkyHub, read/write

**Primary Use** Securely transmit OPC UA, DA, or A&E data between secure networks, without VPNs, cloud hosted (monitoring and supervisory control).

**Configuration**

**OPC Client:** Configure this DataHub as an OPC client to the OPC server.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates connections.

**Data Flow Direction:** Read/Write

**When Connection Initiated:** Override Master’s values with mine

**When Connection Lost:** Mark Master’s data “not connected”

**SkkyHub**

**Tunnel Master:** SkkyHub always acts as a Tunnel master, receiving incoming connections.

**Configuration**

**OPC Server:** Configure this DataHub as an OPC server for the OPC client.

**Tunnel Slave:** Configure this DataHub as a Tunnel slave, so that it initiates connections.

**Data Flow Direction:** Read/Write

**When Connection Initiated:** Get all values from Master

**When Connection Lost:** Mark data here “not connected”
OPC UA Connections

Introduction

OPC Unified Architecture (UA) is the latest standard from the OPC Foundation. Its purpose is to unify the OPC Classic standards of Data Access (DA), Alarms and Events (A&E), and Historical Data Access (HDA) into a single, extensible framework. At the same time OPC UA offers improved networking support, a more sophisticated security model, platform independence, and comprehensive information modeling.

The OPC UA spec allows for implementation across a wide range of hardware platforms and operating systems. The different OPC UA implementations that are possible within this extensible and flexible framework all share a common core OPC UA functionality and interoperability.

The goal of Cogent DataHub’s implementation of OPC UA is to support the functionality most required for industrial process control—secure, real-time data communications. Currently this includes OPC UA Server and OPC UA Client support for Data Access, including Discovery, Address spaces, On-demand, Subscriptions, and Events.

The DataHub offers its OPC UA Server and OPC UA Client support as fully integrated with all other DataHub features. This allows it to act, for example, as an OPC UA - Classic converter, to connect OPC UA to any SQL database, display and access OPC UA data via DataHub WebView, connect to Modbus or Excel, send emails, and support tunnelling OPC DA, server-to-server bridging, data aggregation, redundancy, and more.

Because OPC UA may be new to some DataHub users, this chapter includes background information on OPC UA, such as concepts related to Endpoints and Discovery, Certificates, and Security, while also providing the necessary instructions for configuring the DataHub to act as an OPC UA client or OPC UA server, along with some troubleshooting tips.

Acting as an OPC UA Client

To configure the Cogent DataHub to act as an OPC UA client to OPC UA servers, you can follow these steps:

Click here to watch a video.

1. **Activating OPC UA**
   a. Right click on the Cogent DataHub system-tray icon and choose Properties.
   b. In the Properties window, select **OPC UA**.
c. Check the **Act as an OPC UA Client to these servers** box.

2. **Making the Connection**

Click the **Add** button and fill in the fields in the **Configure OPC UA Data Access Server** Window:

   a. **Connection Name** Choose any name, to be used by the Cogent DataHub to identify the connection. If you leave this blank, the DataHub will assign names incrementally, starting with **OPCUA000**, **OPCUA001**, **OPCUA002**, etc.

   b. For the two next entries there are three different approaches, depending on how this client is able to access the LDS (Local Discovery Server) for the server you want to connect to.

   1. If the LDS appears in the **Discovery Domain** list:

      **Discovery Domain** Choose the OPC UA Discovery Server to which you want to connect. Click the server refresh button ( ) if necessary. A list of available endpoint URLs will appear below.

      **Endpoint URL** Choose the appropriate **endpoint URL** for the protocol that you want to use to connect to the OPC UA Server.

   2. If the LDS does **not** appear in the **Discovery Domain** list:

      **Discovery Domain** Type in the computer name or IP address of the computer running the OPC UA server to which you want to connect. Click the server
refresh button ( ) if necessary. A list of available endpoint URLs should appear below.

**Endpoint URL** Choose the endpoint URL for the protocol that you want to use to connect to the OPC UA Server. When the LDS returns the endpoint, it will probably contain the computer name instead of the IP address that you typed in, above. If so, you will need to edit it, replacing the computer name with the IP address. If the endpoint URL is not there at all, you can type it in, making sure that you enter the complete name.

3. If there is no LDS for the server you need to connect to:

   **Discovery Domain** Type in the IP address of the computer running the OPC UA server to which you want to connect.

   **Endpoint URL** Type in the complete name of the endpoint URL that you want to use.

**Endpoint URL Searches**

For each Discovery Domain, the system will attempt to list its endpoint(s), providing feedback as follows:

- Indicates that the endpoint discovery is in process.
- Indicates that the endpoint discovery has failed.
- Indicates that the endpoint discovery has succeeded.

If a connection has already been configured, and the **Configure OPC UA Data Access Server** Window is opened for editing, the Endpoint URL will first appear as previously configured. The DataHub will then attempt to validate the endpoint, with the status icon changing first to In Process ( ), and then to either Failure ( ) or Success ( ). If at any time you initiate a search by pressing the server refresh button ( ), and the system fails to locate an endpoint URL, then it will leave the **Endpoint URL** entry field empty.

c. **Security Policy** If you are working on an untrusted network, and want to use encryption, then choose a security policy. Otherwise, you can leave this at None.

   The security options available for each endpoint URL are determined by the OPC UA server configuration.

d. **User Token Type** If you want to use authentication, then you can choose a User Token Type. Otherwise, you can leave this at Anonymous.

e. **User Identity** This will change depending on the User Token Type (above), allowing for the entry of a certificate file path or a user name and password, as appropriate.
f. **Always accept invalid server certificate:** This option tells the client to always accept the server certificate, even if the certificate is invalid, or if it changes in the future.

   Selecting this option will disable server certificate verification for this connection, exposing the connection to man-in-the-middle attacks. Use with extreme caution.

g. **Continue to accept server certificate when it expires** allows a UA server certificate to be accepted outside of its valid time window, meaning that expired certificates can continue to be used, and that the UA server and client will stay connected if their system clocks ever get out of synch.

   If you are using the **http protocol** along with a security policy, then the clocks on the UA server and client machine must match within 5 minutes at all times. If this is not possible, you should use either the **opc:tcp** or **https** protocol.

h. **Do not disconnect when the server reports a failed state** By default, if the UA server is in a non-RUNNING state the DataHub disconnects and puts a message in the **Event Log**. Checking this box lets you override that behaviour and maintain the connection to the server.

3. **Connection Test**

   To test the connection, click the **Connection Test** button. The system will open the Connection Test window, and you can watch as it checks the parameters, then creates a channel and session, and then activates the session.

   ![Connection Test Window]

   If there is a problem at any point, the **Message** box will provide some trouble-shooting tips.

4. **Data Transfer**

   There are several options for specifying how the data is to be transferred: If this is the
first time you are configuring an OPC UA client connection, to keep things simple you can keep the default settings for now, and move on to the next step, Selecting Items. Otherwise, you can change the data transfer settings as follows:

![Data Transfer Settings](image)

- **Maximum update rate (milliseconds)** lets you specify an update rate, useful for slowing down the rate of incoming data. The minimum value is 10. This value is also used as the polling time for asynchronous and synchronous reads (see below).

- **Read Method** Choose how to read data from the OPC UA server:
  - **Subscription** The DataHub registers with the UA server for all configured points, to be received on an event-driven basis. Whenever a point value changes, the new value is sent immediately to the DataHub. This option is more efficient than Synchronous Read or Asynchronous Read, and has lower latency than either of them.
  - **Synchronous Read** The DataHub polls the UA server for all configured points on a timed interval (set by the Maximum update rate), and this thread waits for a reply. This option is less efficient than Subscription or Asynchronous Read, and has higher latency than either of them.
  - **Asynchronous Read** The DataHub polls the UA server for all configured points on a timed interval (set by the Maximum update rate), and does not wait for a reply. This option is less efficient than Subscription, and has higher latency.

- **Write Method** Choose how to write data to the OPC UA server:
  - **Asynchronous Write** The DataHub writes to the UA server and does not wait for a response. This provides the highest overall performance.
  - **Synchronous Write** The DataHub writes to the UA server and waits for a response each time. This elicits a quicker response for a given item from the UA server, but results in lower overall performance. This option is useful if the UA server doesn't support asynchronous writes at all, or if it can't handle a large number of them.

- **Monitored Item Queue Size** The maximum number of items between polls that get stored on this server.

- **Maximum Request Item Count** lets you reduce the DataHub default of 500 to
what the server allows. This may be necessary because the OPC UA spec allows a UA server to specify the number of items it will allow per request, which in some cases can be less than 500.

f. **Only transmit GOOD quality data to this server** restricts point updates from the DataHub to the server to only those with "Good" quality.

g. **Do not accept null data from this server** restricts point updates from the server to the DataHub to only those with non-null values.

h. **Create multiple subscriptions using Maximum Request Item Count** The **Maximum Request Item Count** (above) specifies the maximum number of nodes per subscription. With the **Create multiple subscriptions**... option checked (the default), the DataHub will use this number to decide the maximum number of nodes per subscription. However, if this number is small and the total number of nodes is large then the number of requested subscriptions could exceed the subscription count limit of the server. Unchecking this box will solve that problem by putting all of the nodes into a single subscription.

Please see **Advanced** for information about advanced options.

5. **Selecting Items**

You can select all nodes, select nodes manually, or both.

a. **Load All Nodes on Server** With this option you can load all data nodes on the OPC UA server, or filter for groups of nodes.

When you choose this option, the DataHub is configured to provide all data nodes, but not the Server nodes. This is done as a convenience, because in most cases few, if any, Server nodes are needed. To additionally get Server nodes, you can select them manually.

b. **Manually Select Nodes** Select the **Manually Select Nodes** option and press the **Configure Nodes** button.

This opens the Configure Nodes window, where you can specify exactly which nodes you wish to use:
You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. To view sub-branch and leaf items, click the + sign in front of the item to show the children. You can select many items together like this:

a. Expand all of the branches containing points that you want to add.
b. Click the name of the first point (not the check box).
c. Go down to the last point, hold down the Shift key and click the name. All of the names should become highlighted.
d. Press the Space Bar.

That should select all of the highlighted points. It will not select nodes that are not visible.

Selecting just a branch by itself will not include any of its sub-branches or leaves, but selecting a leaf item will automatically include all of its branches.

Checking the box Select variable only data point will ensure that the only nodes you choose are data nodes.

A + in front of an item does not necessarily mean that the item has children. You must click the + sign to find out.

c. Data Domain Name lets you specify the name of the DataHub domain into which the data points will be placed.

6. Click the OK button to accept the configuration (or Cancel to reject it) and close the Configure OPC UA Data Access Server window.
7. Click the **Apply** button in the Properties window to accept the configuration.

The configured client connection will appear in the list, and if the **Act as an OPC UA Client to these servers** option is selected, the DataHub will attempt to connect to the OPC UA server.

### Acting as an OPC UA Server

By default, the Cogent DataHub is configured to act as an OPC UA server. To change this or any default settings, you can follow these steps:

- **Click here to watch a video.**

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **OPC UA**.
3. Ensure that the **Act as an OPC UA Server** box is checked to enable the OPC UA server, or uncheck the box to disable it.
4. These basic UA server settings can be modified if necessary.

Any changes made here will restart the OPC UA server when you click the **Apply** button.

- **Protocol** To disable any of the available protocols, uncheck its box.
- **Port** The port number can be changed by double-clicking it, or by using the **Edit Port** button.
- **Computer Name/IP** You can change the host name or IP address, which will then be integrated into the server URL visible to a connecting client. The default is the host name.
- **Endpoint Name** You can enter an endpoint name for this OPC UA server. This will be integrated into the server URL visible to a connecting client. The default is `CogentDataHub/DataAccess`. 
Some UA clients cannot connect to a UA server unless the server name is left blank. For these cases, the DataHub can be configured with a blank server name as follows:

1. Clear the **Endpoint Name** entry field so that it is blank.
2. Uncheck the **HTTP** and **HTTPS** protocols, as these are not supported when the **Endpoint Name** is blank.
3. Click **Apply** to save the changes.

The DataHub UA server will restart with a blank user name, allowing a UA client to connect to it using a simple Endpoint URL, for example:

```plaintext
opc.tcp://192.168.1.1:52310/
```

Some UA clients may require some or all of the following information about the DataHub OPC UA server:

- **Namespace** http://www.cogentdatahub.com/DataHub
- **Namespace ID** 2
- **ID type** This information should not be exposed to the user.
- **ID** Something like this: `ns=2;s=DataPid:PID1.Mv`. Generally the syntax is `ns=2;pointname`. The namespace is always 2.
- **Type** Typically the canonical type of the node (ID above) retrieved from the server through a client request.
- **Access to data point** The client application developer will need to provide this information, such as read-only or read-write.

5. To make it easy to get started, the OPC UA server is configured with minimal security settings. If you want to modify or enhance security, click the **Advanced** button to open the UA Server Properties window, shown below. Otherwise, you can skip this step.

![UA Server Properties](image)

a. The **General** tab of the UA Server Properties dialog lets you modify the default
Security Policies and User Token Policies for each Server URL.

**Security Policies** Disabling None for opc.tcp and http will require connecting clients to support encryption for these connections. HTTPS is already encrypted, so None need not be disabled for https.

**User Token Policies** Disabling Anonymous will require the connecting client to provide a username/password or certificate to log in.

Putting all this together, below are some suggested settings you can use to secure your OPC UA server.

<table>
<thead>
<tr>
<th>opc.tcp://...</th>
<th>Security Policies</th>
<th>User Token Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Off</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>Basic128Rsa15</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>Basic256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>http://...</th>
<th>Security Policies</th>
<th>User Token Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Off</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>Basic128Rsa15</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>Basic256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>https://...</th>
<th>Security Policies</th>
<th>User Token Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>User Name</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>Certificate</td>
</tr>
</tbody>
</table>

Any changes made here will restart the OPC UA server when you click the Apply button.

b. In the Client Certificate Receiving section, ensure that the Automatically ac-
cept untrusted certificates box is not checked.

Otherwise, all clients will be accepted as trusted.

The **Continue to accept client certificates when they expire** option allows expired certificates to be used, and also keeps the UA server and client connected if their system clocks ever get out of sync. For more information, please see the Advanced feature of the OPC UA Server section.

c. Click the **OK** button to accept the revised configuration (or **Cancel** to reject it) and close the UA Server Properties window.

6. You can use the **Copy Endpoint to Clipboard** button to make a copy of this server's endpoint, if necessary.

7. Click the **Apply** button in the Properties window to accept the configuration.

These are the most common changes you might want to make to the default configuration for the OPC UA server. For other configuration options and more details about the OPC UA server, please refer to the OPC UA Server section of the Properties Window chapter.

## Endpoints and Discovery

An OPC UA connection is initiated by an OPC UA client contacting an OPC UA server. Each client connection is made to the **endpoint URL** that the OPC UA server makes available. To establish a connection, the OPC UA client must either know the endpoint URL of the OPC UA server in advance, or it needs to **discover** it.

To facilitate discovery, an OPC UA server can offer a Local Discovery Server, or **LDS** connection. The LDS is identified by the computer host name, and it always uses port 4840. Thus, as long as the OPC UA client knows the host name of the computer, it can connect to the LDS. Once connected, the LDS provides the endpoint URL for the OPC UA server. Based on that information, the OPC UA client connects to the OPC UA server.

Of course, if the endpoint URL is known, then that can be entered directly in the OPC UA client configuration, and the connection will be made directly.

### Endpoint URL Syntax

The syntax for an endpoint URL is as follows:

```
Protocol://ComputerName:PortNumber/EndpointName
```

*Protocol*

  Defined by the UA Server. The client may only connect using one of the protocols of-
The OPC UA protocols are represented in endpoints as follows:

- **TCP** - opc.tcp
- **HTTP** - http
- **HTTPS** - https

**ComputerName**

The network name, IP address or fully qualified domain name of the computer you are trying to reach. It is dependent on the client-side computer.

**PortNumber**

Usually defined by the UA Server, though you may need to change it if your network connection includes a NAT that is remapping the IP address of the UA Server.

**EndpointName**

Defined by the UA Server.

Examples:

opc.tcp://My-PC:4840/MyComputer/MyUAServer
http://AcmeServer:52601/UADiscovery
https://175.252.04.21:443/Resources/TargetUAServer

**OPC UA Protocols**

OPC UA specifies three data communication protocols. The DataHub supports all of them:

- **TCP** - Binary protocol, highest performance with the smallest overhead and fewest resources, best chance of interoperability with OPC UA servers. Requires TCP port 4840.
- **HTTP** - Binary and XML protocols, lowest performance, usable in Java and .NET environments, extensive tool support, can use TCP port 443.
- **HTTPS** - Binary and XML protocols, a hybrid of TCP and HTTP. More efficient and less overhead than HTTP, wraps a binary payload in HTTPS, can use port 443.

TCP is envisioned as the primary and normal protocol for OPC UA, while HTTPS was implemented to support special cases such as Internet communications. HTTP can be used when web services are necessary and resources are sufficient.

**Security**

OPC UA requires every participating server or client to have a **certificate**. Certificates are authenticated according to one of the four OPC UA Security Tiers:

- **Tier 1 - No Authentication** means an OPC UA client can connect to an OPC UA server with any certificate. Neither the client nor the server authenticates the certificate of the other.
- **Tier 2 - Server Authentication** means that the OPC UA client will check the OPC UA server's certificate against its **trust list** of accepted certificates, and only connect to the
server if its certificate is on the list. However, the server does not check its trust list for client connections.

- **Tier 3 - Client Authentication** means that the OPC UA server will check the OPC UA client’s certificate against its trust list, and only allow a connection from the client if its certificate is on the list. However, the client does not check its trust list for server connections.

- **Tier 4 - Mutual Authentication** means that both the OPC UA server and OPC UA client will check each other’s certificate against their respective trust lists, and the connection will only be allowed if each certificate appears in the appropriate trust list.

The Cogent DataHub supports all of these. The OPC UA server configuration has a Manage Certificates feature where you can search through client certificates and view, accept, reject, or delete them. The OPC UA client configuration allows you to select a user token type (see below), and then enter a username and password, or certificate, as appropriate.

**User Token Types (Log-in types)**

- **Anonymous** The UA server allows any user to connect.
- **UserName** The UA server requires a user name and password.
- **My Certificate** The UA server allows you to use your DataHub’s own certificate. Another
- **Certificate** The UA server requires a certificate other than your DataHub’s own certificate.

**Security Policies**

A security policy determines how an OPC UA server and OPC UA client sign and encrypt messages. The DataHub supports these security policies:

- **None** Supports authentication, but no encryption.
- **Basic128Rsa15** Supports authentication and encryption (AES, key length 128).
- **Basic256** Supports authentication and encryption (AES, key length 256).
- **Basic256Sha256** Supports authentication and encryption (AES, key length 256).

**Certificates**

OPC UA uses certificates to implement security. A certificate is a file that contains identifying information for an OPC server or client, such as the application, machine name, validity period, and so on, along with a private key and public key. When an OPC UA client and server negotiate a connection, they exchange public keys. Each of them uses the key to validate the other’s certificate, and if successful, a connection is made.

There are two ways that certificates can be created:

1. Issued by a Certificate Authority (CA), an independent administrator or organization that verifies the information in the certificate is correct and hasn’t been changed, and pro-
OPC UA Connections

vides a digital signature to that effect.

2. Self-signed, where the administrators of the OPC UA servers or clients need to validate the contents of the certificate themselves.

The Cogent DataHub provides a self-signed certificate, which gets created when the software is first installed. This one certificate is used for both the DataHub's OPC UA server and OPC UA client implementations. The DataHub also supports the ability to use a certificate from a CA, if desired.

Certificates are stored on a system in a certificate store. In Windows, there is a registry-based store called the Windows Certificate Store, and all systems have a directory that stores certificates in a file called the OpenSSL Certificate Store. In addition to these, the DataHub's certificate management feature maintains a private certificate store where you can keep your DataHub-related OPC UA certificates separate from other certificates on your machine.

On the initial connection between an OPC UA client and server, the server checks to see if the client's certificate is available in the certificate store. If not, the server may or may not allow the client to connect. The DataHub can allow a client to connect temporarily by flagging its certificate as "Temporary", which can subsequently be changed to "Rejected" or "OPC UA client" (accepted) as appropriate.

**OPC UA to DA Conversions**

You can use the DataHub to convert data between the OPC UA and OPC DA protocols. The DataHub can connect to any number of OPC UA and/or DA servers or clients, and keeps a copy of all of their registered points in a single, universal data set. Each connected server or client can access that whole data set, and yet sends or receives data value updates in its own protocol.

You can configure OPC UA to DA conversion as follows:

1. Configure the appropriate OPC UA server or OPC DA server connections.
2. Configure the appropriate OPC UA client or OPC DA client connections to access the data from any or all OPC UA or DA servers.
   
   If all of your connections are client-server connections, that may be all that is needed.

   However, if you need to make server-server connections, one more step is necessary
3. Configure DataHub Bridging for any server-server connections.

**Object Model**

Part of the DataHub's job is to convert OPC UA to OPC DA, DDE, and DHTP tunnel protocols. The OPC UA object model cannot be mapped in general to these other protocols, so some simplification is required. The DataHub does not support the complete OPC UA object model. Instead, it simplifies the data model by only looking at hierarchical relationships and converting them all to a simple parent/child "organizes" hierarchy.
Restricting Connections

There is a mechanism in the DataHub to restrict OPC UA clients to selected devices, according to specified DataHub data domains. The clients are distinguished either by user name or by certificate. Configuration is done as follows:

1. In the Security option of the Properties window create a new Group for each device.
2. For each user, create a user name or find the certificate for that user in the UserName list.
3. Add the user to the group for the appropriate device.
4. In the OPC UA Server section of the OPC UA option of the Properties window, choose Advanced.
5. Select Configure Data Domains and change it so that only certain user groups have access to the device data domains.

Once completed, this will require the DataHub to make a separate OPC UA connection to the UA server for each device, putting the data into separate device domains. If it is not possible to make multiple connections to the UA server, then you can make a single connection and use the DataHub Bridging feature to copy the data into separate domains.

OPC UA Test Client and Server

Using a Test Client

To test the Cogent DataHub OPC UA Server using the OPC Foundation's OPC UA Quickstart Data Access Client, follow these steps:

1. Ensure that the DataHub and DataPid are running.
2. Download the OPC Foundation software package from here:
   
   http://cogentdatahub.com/dl/Beta/opc-ua-1.02-.net-sample-applications-setup-336.0-20150630.zip

3. Open the .msi archive to install the software.
4. From the Windows File Menu, go to OPC Foundation, and open the DataAccess Client. This should open the Quickstart Data Access Client:
5. From the Server menu, select Discover, and then press the Find button.

6. Select the address that starts with opc.tcp.

7. Press OK.

8. Click the Connect button in the client window.

You will get an Untrusted Certificate message.

9. Click Yes. You will see an error message, telling you that the connection was not successful. This is normal. The DataHub has rejected the client's certificate, and placed it in the rejected certificate store. At this point you have two options:

   1. You can uncheck the Use Security option in the Quickstart client.

   2. You can go to the DataHub and accept the client certificate by clicking the Accept All button in the OPC UA Server section of the Properties window.

10. Once you have done either or both of these, click the Connect button again. You should now see the DataHub data hierarchy.
To test monitoring, right-click a data point, for example DataPid:PID1.MV, and select **Monitor**. You should see the point and its data appear in the list at the bottom:

11. Do the same for DataPid:PID1.PV and DataPid:PID1.SP. To test writing to the DataHub, right-click a writeable data point like DataPid:PID1.SP, select **Write** and enter a value between 0 and 100. You should see the SP value in DataPid change. Note that if DataPid is running in Auto mode, the value of that point changes automatically every 5 seconds.

**Using a Test Server**

To test the Cogent DataHub OPC UA Client using the OPC Foundation's OPC UA Quickstart Data Access Server, follow these steps:

1. Ensure that the DataHub is running.
2. Download the OPC Foundation software package from here:
   
   ```
   http://cogentdatahub.com/dl/Beta/opc-ua-1.02-.net-sample-applications-setup-336.0-20150630.zip
   ```
3. Open the `.msi` archive to install the software.
4. From the Windows **File** Menu, go to **OPC Foundation**, and open the **DataAccess Server**. This should open the Quickstart Data Access Server:
5. Copy the **Server Endpoint URL** that starts with `opc:tcp`.

6. In the DataHub, go to the **OPC UA** option in the **Properties** window, and click the **Add** button.

7. Choose the **Discovery Domain** for your system, and paste the **Server Endpoint URL** that you copied from the Quickstart Data Access Client into the **Endpoint URL** field.

8. Continue configuring the connection as documented in the section called “Acting as an OPC UA Client”.

**Troubleshooting**

**Endpoint URLs**

**What if the LDS doesn't report back the Endpoint URLs?**

1. Check to ensure that the LDS is running on the server.
2. Check that all firewalls between client and server are allowing access to the LDS port.
   a. Look in the DataHub Event Log for any messages like this that show you the port number of the LDS you are trying to connect to:

   ```
   UA Client: Could not fetch servers from url:
   http://myPC:52601/UADiscovery.
   ```

   b. On the UA server, check the firewall rule. Look for OPC UA Local Discovery Server and make sure it is allowing Inbound connections on that port number and that the firewall rule is set to allow all Profiles (Domain, Private and Public).
3. If you can’t change anything, then you need to resolve this by typing in the Endpoint URL manually.

How do I know what the Endpoint URL is for the UA server?

The syntax for an endpoint URL is as follows:

```
Protocol://ComputerName:PortNumber/EndpointName
```

**Protocol**

Defined by the UA Server. The client may only connect using one of the protocols offered by the server. The OPC UA protocols are represented in endpoints as follows:

- **TCP** - opc.tcp
- **HTTP** - http
- **HTTPS** - https

**ComputerName**

The network name, IP address or fully qualified domain name of the computer you are trying to reach. It is dependent on the client-side computer.

**PortNumber**

Usually defined by the UA Server, though you may need to change it if your network connection includes a NAT that is remapping the IP address of the UA Server.

**EndpointName**

Defined by the UA Server.

Examples:

- opc.tcp://My-PC:4840/MyComputer/MyUAServer
- http://AcmeServer:52601/UADiscovery
- https://175.252.04.21:443/Rsources/TargetUAServer

Please see the section called “Endpoints and Discovery” for more information.
Error Messages

Errors related to Endpoint URL

- **Error:** A requested configuration was not found.
  **Possible cause:** Endpoint Name defined in the Endpoint is incorrect.

- **Error:** BadTcpInternalError: Error establishing a connection.
  **Possible cause 1:** The server is not running.
  **Possible cause 2:** The port number defined in the Endpoint is incorrect.

- **Error:** No such host is known.
  **Possible cause:** The computer name, IP address or domain name defined in the Endpoint is incorrect.

- **Error:** BadTcpEndpointUrlInvalid: The Server does not recognize the QueryString specified.
  **Possible cause:** The protocol defined in the Endpoint is incorrect or mistyped.

- **Error:** No such host is known.
  **Possible cause:** The IP address (or computer name or domain name) defined in the Endpoint is incorrect. This error can also occur if you are using an IP address in the Discovery Domain field to return the list of Endpoint URLs from the server. The endpoints returned by the server will include the server’s computer name rather than the IP address, so you need to edit the endpoint URL to replace the computer name with the IP address like this:

  opc.tcp://192.168.3.151:51310/OCS4DDataLink/DataAccess

  The **Endpoint URL** is made up of these parts:

  protocol://computer name:port number/endpoint name

Errors related to Security Policy (Encryption)

- **Error:** BadSecurityChecksFailed:
  An error occurred while verifying security.

**Configuration:** Protocol = opc.tcp, Security Policy = Basic128Rsa15 or Basic256, Token Type = any.

**Possible cause:** This means your connection succeeded but the certificate being used to enable encryption has been rejected by the server. The UA Server needs to accept this certificate in order for this connection to succeed.

- **Error:** An unsecured or incorrectly secured
fault was received from the other party. See the inner FaultException for the fault code and detail.

**Configuration:** Protocol = http, Security Policy = Basic128Rsa15 or Basic256, Token Type = any.

**Possible cause:** This means your connection succeeded but the certificate being used to enable encryption has been rejected by the server. The UA Server needs to accept this certificate in order for this connection to succeed.

**Errors related to User Token Type (Authentication)**

- **Error:** 'CN=Cogent DataHub Data Access Server, DC=WIN10VM' is not a trusted user certificate.

  **Configuration:** Protocol = any, Security Policy = any, Token Type = certificate.

  **Possible cause:** This means your connection succeeded but the certificate being used to authenticate has been rejected by the server. The UA Server needs to accept this certificate in order for this connection to succeed.

- **Error:** BadIdentityTokenRejected:
The user identity token is valid but the server has rejected it. Possible cause, the user token is not supported on the server or my certificate is not trusted on the server.

  **Configuration:** Protocol = any, Security Policy = any, Token Type = certificate.

  **Possible cause:** This means the user name or password used to authenticate is incorrect.

- **Error:** BadUserAccessDenied:
  User does not have permission to perform the requested operation.

  **Configuration:** Protocol = any, Security Policy = any, Token Type = User Name

  **Possible cause:** This means the user name or password used to authenticate is incorrect.

**Errors related to possible network problems**

These will typically involve a long timeout delay before the error message is displayed.
• Error: There was no endpoint listening at http://win8vm:51311/OCS4DDataLink/DataAccess/discovery that could accept the message. This is often caused by an incorrect address or SOAP action. See InnerException, if present, for more details.

**Configuration:** Protocol = any, Security Policy = any, Token Type = any.

**Possible cause:** This may be caused by a firewall between the client and the server not allowing access through the specific port used by the protocol you have chosen. Add new firewall rules as needed to allow access.

• Error: The underlying connection was closed: A connection that was expected to be kept alive was closed by the server.

**Configuration:** Protocol = any, Security Policy = any, Token Type = any.

**Possible cause:** This may be caused by a firewall between the client and the server not allowing access through the specific port used by the protocol you have chosen. Add new firewall rules as needed to allow access.

• Error: BadCommunicationError: A low level communication error occurred.

**Configuration:** Protocol = any, Security Policy = any, Token Type = any.

**Possible cause:** This may be caused by a firewall between the client and the server not allowing access through the specific port used by the protocol you have chosen. Add new firewall rules as needed to allow access.

**Other errors**

• Error: BadInvalidTimestamp: The timestamp is outside the range allowed by the server.

**Cause:** The system clock on the client computer is outside the acceptable range defined by the UA server
Bridging

Introduction

*Bridging* means connecting two data servers together so they can access each other's data, even when neither of them can act as a client.

In addition to bridging two servers on a single computer, the Cogent DataHub offers advanced bridging capabilities that let you:

- Bridge servers over a network connection, by *tunnelling*.

- Bridge between any number of servers, through aggregation.

- Scale, convert, or normalize the data as it is bridged from one server to the other, with built-in *linear transformations*.

- Define even more complex relationships between points in code using *DataHub Scripting*.

You can *configure the bridges* you need using the Bridging option in the Properties Window.

Configuring Bridges

It is easy to configure the Cogent DataHub to bridge existing points—just point and click. If necessary, you can quickly configure *linear transformations* and specify the direction of
data flow of the bridge. And should you want to create a new point for a bridge, it's just few more clicks of the mouse. All configuration and any changes are done on the fly, taking effect as soon as you click the Apply button.

Click here to watch a video.

Point-to-point configuration

1. With the DataHub running, right click on the Cogent DataHub system-tray icon and choose Properties.

2. In the Properties window, select Bridging.

3. Click the Configure Bridges button. The Bridging Configuration window will open.

![Bridging Configuration Window](image)

The three top panes in this window correspond to the three basic steps in making the configuration: specify a source, a destination, and any desired transformations. The horizontal pane across the bottom shows the bridges that exist on the system.

4. From the tree diagram in the Source panel, select a source point that you want to bridge.
For example, if you have the DataSim program running, you can select the point Sine in the DataSim data domain. The name of the point gets automatically entered in the field at the top of the panel. Alternatively, you can type the name of the point in the entry field.

5. In the tree diagram in the Destination panel, select a destination point.

When you select a destination point, its name gets automatically entered in the field at the top of the panel. Or you can type the name of a point in the entry field.

6. Specify direct copy or transformation.

To make a direct copy, just leave the default Direct copy selected. To make a linear transformation, select Linear Transformation or Linear Range Mapping and enter the appropriate data, as explained in the section called “Making transformations” below.

7. Determine which direction you want the bridge to apply.
• Select **Forward** to change the destination point when the source point changes, but not change the source when the destination changes. If you select **Force consistency** with this option, and if the destination point gets changed for some reason, then the DataHub will attempt to force its value to be consistent with the source point value.

• Select **Inverse** to change the source point if the destination point changes, but not vice-versa.

Selecting **Inverse** will apply the inverse of the transformation, as explained below.

• Select both **Forward** and **Inverse** for a bidirectional bridge, where either point changes whenever the other point changes. This combination will deselect **Force consistency** to eliminate the possibility of conflicting behavior.

8. Click the **Apply** button to create and activate the bridge. The DataHub will create the bridge and update the bridged points immediately.

9. In the bottom panel you can see all the bridges that exist in the system, and the significant information about them.

If you click on a transformation, the source point, destination point, and transform information get displayed in their respective panels. Use the check box at the front of each bridge to activate or deactivate it.

**Making transformations**

1. Specify the type of transformation by clicking one of the three radio buttons:
• **Direct copy** makes no transformations. It just copies the point.

• **Linear Transformation** lets you multiply by one value and add another value, such as in the equation \( y = mx + b \) where the destination point is \( y \), the source point is \( x \), the **multiply by** value is \( m \), and the **then add** value is \( b \). For example to transform a Celsius source point to a Fahrenheit destination point, you would multiply by 1.8 and add 32, or

\[
\text{Fahrenheit} = (1.8 \times \text{Celsius}) + 32
\]

If you have selected the **Inverse** direction for a transformation, you will get the inverse of the transformation. In this example, you would get a conversion from Fahrenheit to Celsius, or the results of this equation:

\[
\text{Celsius} = (\text{Fahrenheit} - 32) / 1.8
\]

As an alternative to entering transformation values, the DataHub also offers **Linear Range Mapping**.

• **Linear Range Mapping** lets you enter a range for the source and destination, and the DataHub automatically calculates the corresponding linear transformation. For example, to create the same Fahrenheit to Celsius transformation, you could use the defaults of 0 and 100 for the **Min** and **Max** of the source point. Then you would enter 32 and 212 for the **Min** and **Max** of the destination point. As soon as you make these entries, the correct values get entered automatically in the **Linear Transformation**.

When you use linear range mapping, you can limit the transformed value to the maximum and minimum by checking the **Clamp** boxes. The clamps get applied to the point being changed, i.e., to the destination point for forward direction, to the source point for inverse direction, and to both points for bidirectional bridges.

2. If you want to save this transformation for future use, click the **Store** button at the top of the Transformation panel, and enter a name in the box that pops up.
Once stored, the transformation will become available by name in the drop-down list.

3. To load a transformation, simply select its name from the drop-down list.

Creating New Points

A special feature of the Cogent DataHub allows you to create new points. Combined with the ability to create new data domains, this lets you create:

- **Personalized data sets** for different users or groups.
- **Aggregated data sets** that combine selected data points from different servers and serve it up from a single OPC server (i.e. the DataHub).
- **Scaled data sets** that apply one or more transformations to a subset of the data.
- **Temporary data sets** for testing and demonstration purposes.
- **Any combination** of the above.

You can create a new point in the Source or Destination panel of the Bridging Configuration window by typing in a new point name in the entry field.
And with a few mouse clicks you can quickly create bridges to new destination points.

![Bridge diagram]

Just click on a source point, click on a data domain, and then click **Apply**. A destination point with the same name as the source point gets created automatically in the data domain you chose, with the current transformation applied.

You might want to create special data domains for holding sets of destination points. For more information about data domains, please refer to the section called “Data Domains”.

### Configuring Many Bridges

If you only have a few data points to bridge, then using the Bridging option in the Properties window, as described in the section called “Configuring Bridges” works very well. However, there is a faster way to configure many bridges, using a custom configuration file. Here’s how:

1. First configure at least one bridge using the Bridging interface so you can see the syntax used in the DataHub configuration file.

2. When that’s done, open the `Cogent DataHub.cfg` file, found here:

   ```
   C:\Users\{username}\AppData\Roaming\Cogent DataHub\Cogent DataHub.cfg
   ```

   Find the section of the file that looks like this:

   ```
   ;;; Point-to-point Bridging
   (bridge "domain:pointname" "domain:pointname" 257 1 0 0 100 0 100)
   ```

   This is how each bridge gets stored in the DataHub's configuration file, as a **bridge** command that gets called when the DataHub starts up. Your custom configuration file needs to contain one line like this for each bridge that you configure.

3. Create a custom configuration file, as described at the end of the section called “Configuration Files”. The content of the file should look like this:

   ```
   ;;; My Point-to-point Bridging List
   (bridge "SourceDomain:Point1" "DestinationDomain:Point1" 257 1 0 0 100 0 100)
   (bridge "SourceDomain:Point2" "DestinationDomain:Point2" 257 1 0 0 100 0 100)
   ```
You should not try to edit the Cogent DataHub.cfg file. Put all of your custom bridge configuration into the custom configuration file.

4. Configure the DataHub to load your custom configuration file from the Scripting option in the Properties window, as described at the end of the section called “Scripting”.

5. Restart the DataHub.

6. Open the Bridging option in the Properties window, and click the Configure Bridges button. You should see all of the configured bridges.

7. Check the Data Browser. You should see all of the configured bridged points.

Bridging Scenarios

Here are some of the most commonly used bridging scenarios.

Bridging Local Servers

The most common scenario for OPC bridging is connecting OPC servers on a single machine. This can be used to create connections over various fieldbus protocols. Any fieldbus connected to an OPC server can be bridged to any other.
The bridge is **configured** as a direct link, or it might incorporate **linear transformations**. It can have a forward or inverse direction, or be bidirectional.

**Bridging Remote Servers**

A special bridging application that the Cogent DataHub makes possible is to bridge remote servers. This is effectively bridging combined with **tunnelling**.

---

**You can use this to create a software bridge between remote pieces of hardware. For example, consider this situation:**

---

Suppose you need to control a motor remotely. An on/off switch and status indicator light are connected to PLC "A" in one location. The motor itself and a rotational sensor are connected to PLC "B" in another location. Both PLCs have OPC servers connected to copies of the Cogent DataHub.

To make the connection, you would need to program the PLCs for these data points:

<table>
<thead>
<tr>
<th>Point</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.switch</td>
<td>- Changes value to 1 when switch is on, 0 when switch is off.</td>
</tr>
<tr>
<td>A.indicator</td>
<td>- Turns on light when value is 1, switches off light when value is 0.</td>
</tr>
<tr>
<td>B.motor</td>
<td>- Starts motor when value is 1, switches off motor when value is 0.</td>
</tr>
<tr>
<td>B.sensor</td>
<td>- Changes value to 1 when shaft is turning is on, 0 when shaft is not turning.</td>
</tr>
</tbody>
</table>

then bridge them in the DataHub like this:

<table>
<thead>
<tr>
<th>On</th>
<th>Source</th>
<th>Destination</th>
<th>Fwd</th>
<th>Inv</th>
<th>Multiply</th>
<th>Plus</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>A.switch</td>
<td>B.motor</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>A.indicator</td>
<td>B.sensor</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
As when bridging local servers, bridging remote servers can also bridge different fieldbus protocols.

**Creating Data Sets**

A third common application for Cogent DataHub bridging is creating custom data sets. You can select certain data points from several OPC servers and group them into different data domains depending on the user. This has two advantages:

- It simplifies configuration and use.
- It reduces bandwidth across the network.

For example, suppose you needed to get OPC data to three sets of users: accounting department staff, executives on the road, and an OPC server at a remote plant. You could declare three new data domains, such as Accounting, Executive, and Remote. For each data domain, you would then create new points specific to the needs of those users. With this approach, you can limit the data set and customize it for each recipient.

**Bridging to Excel**

The Cogent DataHub makes it easy bridge data from an Excel spreadsheet to an OPC server.

Simply connect the DataHub to Excel, then configure the bridges. To get Excel data into an OPC server without changing any names in Excel or any OPC point names, follow these steps:
1. **Define a DDE Item in the DataHub** for the Excel point. For example, the cell A1 in a spreadsheet would become DDE Item r1c1 (Row 1, Column 1).

![DDE Item Definition](image)

2. **Configure a point-to-point bridge** using the Excel point as the source and the OPC point as the destination. If we used our OPC server A and the point name switch from the example above, the bridge would look like this:

   ![Bridge Configuration](image)

   This bridge would allow you to turn the switch on and off (and thus control the motor) from an Excel spreadsheet.

### Bridging and Tunnelling

OPC bridging with **tunnelling OPC** means linking OPC servers across a network. Here's how it's done:

1. **Bridging two OPC servers** over a network requires a tunnelling connection.

   ![Bridging Diagram](image)

   This scenario involves setting up the DataHub on both machines to **act as OPC clients** to the respective OPC servers. The DataHubs then interface with each other over a TCP tunnelling connection. Configure the DataHub on the machine with the most uptime to be the **tunnelling master** and the other DataHub to be the **tunnelling slave**. For simplicity sake, we recommend that you **configure all bridges** on one machine.

2. **Bridging two OPC clients** over a network also requires a tunnelling connection.
This scenario involves setting up the DataHub on both machines to act as OPC servers to the respective OPC clients (see above). The DataHubs then need to interface with each other over a TCP tunnelling connection. Configure the DataHub on the machine with the most uptime to be the tunnelling master and the other DataHub to be the tunnelling slave. For simplicity sake, we recommend that you configure all bridges on one machine.
Using Redundancy

Redundancy in a process control system means that some or all of the system is duplicated, or redundant. The goal is to eliminate, as much as possible, any single point of failure. When a piece of equipment or a communication link goes down, a similar or identical component is ready to take over.

For data communications, redundancy means sending data from a single data source over two separate paths, and presenting it to a user as a single data set. The two paths may include redundant PLCs, redundant OPC servers, redundant networks, and so on. The data flow is typically from data source to data user.

The DataHub’s Redundancy feature operates at the point where the two data paths meet. It monitors both data streams, and determines which of them should be used from that point onwards. The choice between data stream depends on Redundancy criteria that you configure. Data from the two incoming streams and the outgoing stream is maintained in the DataHub in separate data domains.

Configuring the Redundancy feature is explained in the Redundancy section of the Properties window chapter of this book. This Using Redundancy chapter will help you get the most out of this DataHub feature.

Typical Scenarios

Redundancy Scenario 1 - No redundancy

- One process, one client, one computer
- Single data path (PLC and OPC server)
**Redundancy Scenario 2 - Redundancy with one computer**

- One process, one client, one computer
- **Redundant** data paths: 2 PLCs and 2 OPC servers

**Redundancy Scenario 3 - Networked redundancy**

- **Redundant** data paths
- **Tunnel** connections
- Two OPC server machines, one OPC client machine
Redundancy Scenario 4 - Networked, multiple client redundancy

- Redundant data paths
- Tunnel connections
- Two OPC server machines, two OPC client machines

Configure the switch

Switching Criteria - Domains

Generally speaking, the criteria for switching between redundant data streams should reflect expectations from the plant or process. The criteria for an invalid domain apply to identical points in each of the two input data domains. When that criteria is met in one domain, it is no longer considered valid, and the DataHub makes the switch.

Key concepts:

- **Test criteria are for the whole input domain.** Whenever the criteria you enter are met, then the whole domain is considered invalid. For example, your OPC connection status may be Running, but if the data quality for your test point(s) meets your criteria for invalid data, then that entire domain will be considered invalid, triggering a switch to the other domain.

- **Whenever both input domains are invalid** then the DataHub sets the quality of the points in the redundancy output domain to Not Connected.
Switching Criteria - Points

For any point in the domain means that your entire data set is considered invalid if even one data point meets your test criteria. Instead, consider using a sentinel point or pattern.

For this point lets you choose a sentinel point or specify a pattern to act on behalf of the whole domain.

- Sentinel Point: A sentinel point is a single data point that acts as a representative, or "sentinel", on behalf of the entire data domain. This can be useful if you want to allow some bad quality data in the input domain, but where one point in the domain will always have good quality whenever the data connection is valid. A sentinel point can also be used to test for connectivity, by testing for the Not Connected status of OPC.

To configure a sentinel point, choose For this point and name one data point that you trust (perhaps a point generated by a PLC rather than a device). This point needs to exist in both of your input domains. The point name should be entered with no domain prefix.

- Pattern matching: If you select Treat the point name as a pattern then the point name acts as a globbing pattern, using matching rules. For example, if you specify a pattern of *.Setpoint then you would be indicating that the data domain is invalid if any data point ending with .Setpoint is invalid.

Valid and Invalid OPC Qualities

The OPC spec has two general categories for valid and invalid qualities, where only Good and Local Override are valid. The rest of the OPC qualities, such as Initializing, Out of Service, Sensor Failure, Not Connected, Bad, and so on are considered invalid for different reasons. If you want to test for all of these, and switch whenever there might be an interruption to the data stream, or a change in data quality, you can choose not equal to and Good for Data quality is in the Input Domain configuration.

There are two things you'll need to consider:

1. If a point's quality changes to Local Override it will cause the system to switch. This rarely happens unless someone enters a value for a point in the DataHub's Data Browser.

2. If you choose this setting for all points, then the system will switch whenever any of the points on the current source is not Good. Consider instead testing a sentinel point or pattern, as explained above.
Checking value AND quality

The interface does not allow for checking both the value of a point and its quality. You can use this script to create a synthetic point that reacts to both, and lets you check them.

Example Script RedundancyState.g

```plaintext
/* Creates a state point that combines both quality and value to produce a single output value that can be used by a redundancy pair to determine input domain validity. The computeState method can be as complex as you like, so long as it finally writes a 0 or 1 to the state point. */

require ("Application");

class RedundancyState Application
{
    // The state point to be used for configuring the redundancy pair
    statePoint = "RedundancyState";

    // The point in the input data set to check for quality and value
    sentinelPoint = "WinAC1500.Memory.master";

    // The names of the input domains.
    domains = [ "HPC_1", "HPC_2" ];
}

method RedundancyState.computeState (pointSymbol)
{
    local info = PointMetadata(pointSymbol);
    local isBad = (info.quality != OPC_QUALITY_GOOD || info.value == 0);
    datahub_write(string(info.domain, ":", .statePoint), isBad ? 0 : 1);
}

method RedundancyState.constructor ()
{
    local pointName, pointSymbol;
    with domain in .domains do
    {
        pointName = string(domain, ":", .sentinelPoint);
        pointSymbol = symbol(pointName);
        datahub_command(format("(create %s:%s 1)", domain, .statePoint), 1);
        datahub_command(format("(create %s:%s 1)", domain, pointName), 1);
        .OnChange(pointSymbol, `(@self).computeState(#@pointSymbol));
        .computeState(pointSymbol);
```
Troubleshooting

A good first step in troubleshooting redundant DataHub connections is to check the Event Log. There you may see error messages related to a configured redundancy pair. In addition to this, the Redundancy feature offers some useful troubleshooting options.

Diagnostic Points

In the Redundancy Configuration dialog you can specify diagnostic points that will get added to the root of the output data domain. These points let you monitor the status of the redundancy pair.

These points will continue to update even when redundancy is disabled. This gives you a way to examine the control bits to determine that the redundancy pair is not currently enabled.

There are two kinds of diagnostic points.

1. Status and Control Data Points: Assigning names to these points will add them to the root of the output domain. If you start each name with a unique character or two, like a double underscore (___), the points will all sort together at the beginning or end of the point list, making them easier to find and view together.

   - Point for current source number: When set, this point indicates which source domain is currently in use. Its value will be one of these:
     - 0 for none (neither input domain passes the validity check).
     - 1 for source domain 1.
     - 2 for source domain 2.
   - Point for current state of domain 1: This point’s value will be one of:
     - 0 if the data in domain 1 is not valid, according to the validity settings.
     - 1 if the data in domain 1 is valid.
   - Point for current state of domain 2: Same as above.
   - Point for preferred source number: This point’s value will be one of:
Using Redundancy

- 0 if there is no preferred source.
- 1 if input domain 1 is the preferred source domain.
- 2 if input domain 2 is the preferred source domain.

2. **Input Statistics Data Points:** In the Options section of the Redundancy Configuration is an option to Create data points showing input statistics, which is off by default.

   ![Create data points showing input statistics]

   If this is turned on it will create the following data points in the output data domain:
   - **Input1ValidCount** shows the number of valid points in input domain 1.
   - **Input1InvalidCount** shows the number of invalid points in input domain 1.
   - **Input1UninitializedCount** shows the number of uninitialized points in input domain 1. Uninitialized points are ones that have not yet received a value from any source. Changing the radio-button selection from Invalid to Valid tells the redundancy engine to ignore such points when evaluating the validity of the domain. Selecting Normal treats them like all of the other points in the domain.
   
   Similar points are created for Input2.

**Switchover Pause**

Any pause during switchover is the time it takes to confirm that the source domain has become invalid. Normally this is the connection timeout period. Until that timeout occurs, the DataHub still considers that domain to be valid. The status for those points won't change to Not Connected until the DataHub is aware that the connection has been lost. To reduce the time of the switchover pause you can reduce the connection timeout on the sources, or configure Data Flow Detection for this redundancy pair.

**Multiple Redundant Pairs**

The Redundancy feature can accommodate any number of redundant pairs. Typically this means separate pairs, like A1/A2, B1/B2, C1/C2, and so on where A, B, and C are the data sources for the redundant pairs. To do this, each redundancy pair must have a unique output domain. You cannot send data from multiple redundancy pairs into a single output domain.
It is also possible to configure redundancy for multiple identical redundant sets coming from a single source of data. For example, you may want to configure redundancy for 4 data paths from a single source, where you have 2 redundant servers running on 2 redundant networks. Redundancy in this scenario would need to be resolved in two stages.

• In the first stage you could resolve the redundant networks into a pair of outputs, one for each server. At the second stage you would resolve the redundant servers to a single output.

• Alternatively, you could resolve the redundant servers to one output for each network, and then resolve the redundant networks into a single output.

Memory Limitations

There are no programmed-in limits on the number of domains or redundancy pairs. The practical limit is typically the number of data points that the DataHub can handle before running out of memory. The total data point count is the sum of the data points counts in each domain.
The number of domains is the total of all input and output domains on the DataHub resolving the redundant connections. On the simplest redundant system that would be 2 inputs and 1 output, for a total of 3. This means you need to ensure you have enough memory for 3 times the number of points as are in one of the source domains.

In the more elaborate scenarios shown above, at the first level you have 4 inputs and 2 outputs, or 6. At the second level the two input domains have already been counted (as outputs on the previous level). You just need to add the final output domain for a total of 7. So on the DataHub that is resolving the redundancy, you need 7 times the amount of memory that a single domain would require.

### Warm Standby

It is possible to use the DataHub’s Redundancy feature to support warm standby, by using a DataHub script. Since we perform redundancy at the data level instead of the connection level, the DataHub does not naturally know which connections to start and stop when it switches input domains. Consequently you need a script that will watch the redundancy state and turn on or off the connections based on separate knowledge about how the connections match up with the data domains.

Below is an example script that shows how to set up warm standby redundancy. It requires you to first configure redundancy through the interface, and then re-enter some of the same information again. For example, here is an OPC configuration, showing two connections named `OPC000` and `OPC002` feeding data into domains `input1` and `input2`:

1. Configure the OPC servers.

2. Configure a redundancy pair labeled, for example, `RED001`.

3. This redundancy pair **must** include configuration for the four status points. You can name them whatever you like:
4. The script needs to be edited so that it contains the configured information about the redundancy pair, including the **Label**, **Source Domain 1**, **Source Domain 2**, **Output Domain** name as well as all four **Status and Control Data Point** names.

   ```
   method WarmStandby.constructor ()
   {
     local redconf = new DomainBridge();
     redconf.SetApplication(self);
     redconf.Attach("RED001", "input1", "input2", "output", 0, 0, nil, 0, nil,
     "CurrentSource", "State1", "State2", "Preference");
     redconf.OPCWarmStandby("OPC000", "OPC002");
   }
   ```

   The rest of the information in the `Attach` call will be ignored. Once we have attached to the redundancy pair, we can tell the script which OPC connections correspond to the input domains. These are provided as the OPC connection label corresponding to each input domain.

   When you run the script, it will watch the status points and determine which OPC connection should be active and which should be inactive based on the status of the data.

   **Example Script WarmStandby.g**

   ```
   require ("Application");
   require ("DomainBridgeSupport");

   class WarmStandby Application
   {
     }

   method WarmStandby.constructor ()
   {
     local redconf = new DomainBridge();
     redconf.SetApplication(self);
     redconf.Attach("RED001", "input1", "input2", "output", 0, 0, nil, 0, nil,
     "CurrentSource", "State1", "State2", "Preference");
     redconf.OPCWarmStandby("OPC000", "OPC002");
   }

   ApplicationSingleton (WarmStandby);
   ```
**Special Cases Q&A**

**Different data sets**

**Q:** Will the Redundancy feature work properly if one of the two input domains has points that the other input domain doesn't have? If that is the case, will it throw some kind of error message?

**A:** Redundancy will work. If there are points in input 1 that are not in input 2 then they will appear in the output when input 1 is active, and get marked as Not Connected when input 2 becomes active. If both input 1 and input 2 become active at some point in a session then the output will be the union of input 1 and input 2, with the points that only appear in the inactive input marked as Not Connected.

If you want to use redundancy, then you need to ensure that the two input domains contain points with identical names. If the point names are not the same from the two sources then you need to create a synthetic input domain and bridge all of the points from one of the source domains into it, changing the point names as part of the bridge.

**Writing back to redundant sources**

**Q:** Writing to the output domain only updates the currently active redundancy connection. Why not both?

**A:** In general there is no expectation that both pathways are always available. The inactive source should be receiving updates to its points from the underlying process, not the DataHub.

If both input domains are available, writing to both of them would cause two write events to the data source, which can produce strange results. Consider a boolean value in a PLC that acts as a rising edge trigger. If the PLC responds to the write by resetting the boolean then writing to both connections will cause the trigger to run twice. This is considered undesirable behaviour.

This has special implications for OPC A&E. If you are reading from two different A&E servers then you have to accept that one of the two servers will not receive an acknowledgement from the A&E client. This is because it may not be connected, or that it is connected but not active. If the inactive A&E server becomes active, the A&E client may see some acknowledged events become unacknowledged as part of the fail-over and will need to re-acknowledge them. Both A&E servers should show the A&E conditions to be in the same state (aside from ACK) as they should be receiving their state information from the same underlying process.

**Replicating a tree hierarchy**

**Q:** How can I get the data hierarchy in my input domains into my output domain?
A: A hierarchy in the output domain is created when an input domain becomes valid. There are two settings in the redundancy configuration:

- **Never copy the data model to the output domain**
- **Always copy the data model when switching**

If neither of these options are checked (the default) the data hierarchy gets copied to the output domain whenever the input domains switch. If neither input domain becomes valid then all points in the output domain will have bad quality and will be in a flat organization.

If the first option is checked then the hierarchy will not appear in the output domain.
Write to a Database

Introduction

The Cogent DataHub can write data to any ODBC compliant database or text file.

With this feature of the DataHub you can:

• Log data to any ODBC-compliant database, such as MS Access, MS SQL Server, MySQL, Oracle, and many more.
• Log from any data source connected to the DataHub.
• Log to existing database tables, or create new tables as necessary.
• Log data to a text file using the LogFile.g script.

To write data from a database into the Cogent DataHub, please see Query a Database.

The Cogent DataHub’s ODBC Data Logging interface provides an easy way to connect to a DSN, create or select a table, assign data point properties to table columns, and assign a trigger and conditions for logging points. The fastest way to learn how to use the interface is by watching the web-site video or by using the Quick Start.

Quick Start

Here’s how to configure the Cogent DataHub to write data to a database of your choice. We use a data point from the DataSim program in this example but you can just as easily use your own data point.

Click here to watch a video.

Open the ODBC Data Logging window

1. In the Cogent DataHub Properties window, select Data Logging.
2. Click the **Configure** button.

![Database Configuration](image)

This opens the ODBC Data Logging window, shown below.

The **Data Logging Configuration** interface is explained in detail in the section called "Configuring the Queue, Store and Forward". For now, you can use the defaults.

### Connect to the Database

1. Select the **1. DSN** tab. A *DSN* is a Data Source Name. Windows uses this name to identify the database you want to connect to.

![Write to a Database (ODBC)](image)

2. From the drop-down box, select a DSN. If you do not have any DSNs, or you wish to create a new DSN, you can do this by opening the DSN Administrator. Please refer to **Setting up a DSN** for more details.

3. Enter the **user name and password** (if required), and click the **Connect** button. A "Connected to . . ." message should appear in the message box. If you get an error message in the box, consult your system administrator.

### Configure a Table

1. Select the **2. Table** tab.
2. Start the DataSim program if it isn’t already running, and ensure that it is connected to the DataHub.

3. In the Table name field type: datasimtest.

4. Click the Add Column button, type: POINTID in the pop-up dialog, and then click OK.

5. Click under the POINTID label in the Item row.

6. Select <key> from the drop-down list. Notice after you make your selection that the word counter gets entered automatically for the Data Type.

7. Click the Add Column button and enter the name PTNAME.

8. In the point-picker list on the right, expand the DataSim data domain and select the point named Sine.

9. In the PTNAME column, click in the Item row and select <point>. The full name of the point, DataSim:Sine, should appear.

   for Property in that column select name.
   for Data Type select varchar (or the equivalent).
   for Data Size the system might have entered a value for you. If not, type in a value like 64 and click Enter.

10. Click the Add Column button again and add the column name PTVALUE. Then make these entries:
for Item select <point>. (The point name, DataSim:Sine, should appear.)
for Property select value.
for Data Type select number (or the equivalent).
for Data Size, depending on your database, you might not be able to enter anything. If you are able to make an entry, you can type in a number of bytes and click Enter.
The entry fields should now look similar to this:

![Database Configuration Screen](image)

11. Click the Create Table button. If successful, you have now created a new table in the database specified by your DSN. You can open your database program and view it to verify. If you get an error message, check your entries above carefully to ensure they are compatible with your database. For example, some databases will not allow spaces or special characters in table and column names.

Once the table is created, you cannot add any more columns. However, you can delete the table using the Delete Table button. This will delete the table from the database, but all of your entries will remain in the entry fields. You can then add more columns if you wish, and recreate the table. You can easily rename, insert, or delete a column by right-clicking on the column name for a pop-up menu. For more information about creating and modifying tables, please refer to the section called “Configuring a Database Table”.

12. When you have the table the way you want it, go down to the Configured Actions box and click the Create button.

![Configured Actions Screen](image)

A new configured action should appear in the list. For more information about configured actions, please refer to the section called “Configured Actions”

Next, to get the DataHub to write the data, you need to assign a trigger.

**Assign a Trigger**

For this example, we will trigger the action whenever the DataSim:Square point changes.
1. Select the **3. Trigger** tab.
2. From the point selector, expand the DataSim data domain and select the point Square.
3. Click the + button to the right of the **Point Name** field. The point name DataSim:Square should fill in for you.

![Image of Write to a Database interface]

You can choose any point for the trigger, including the point that gets written, such as DataSim:Sine in our example. For more information about triggers, please refer to the section called “Assigning a Trigger”.

4. In the **Configured Actions** box, make sure that the configured action you just created is highlighted. If not, click on it to highlight it. Then click the **Modify** button.

![Image of Configured Actions interface]

Your configured action should now display the DataSim:Square point as your **Trigger**.

5. Click the **Apply** button to activate the configured action.
6. Open the **Script Log** to check for error messages and ensure that your data is being written successfully. Each write action to the database gets logged here. You can also verify the writes by querying the database itself.

You have just configured an action that logs the name and value of the Sine point in the DataSim data domain whenever the value of the Square point changes. Now you can create tables to log your own data, with triggers based on points or timers.

The remaining sections in this chapter explain the interface in more detail, and introduce the option of setting specific conditions for logging, if desired.
Configuring the Queue, Store and Forward

The DataHub maintains an in-memory transaction queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. The **Maximum transaction queue** option lets you modify the depth of this queue, with a default of 100 messages. The **Reconnection delay (s):** option specifies the number of seconds before a reconnect is attempted if the ODBC connection is broken. And the **Show diagnostics in the Script Log** option lets you view messages about the connection in the Script Log.

Store and Forward

The term *store and forward* refers to a type of database connection where the data is stored locally to disk and then later forwarded to the database. The Cogent DataHub performs an advanced form of store and forward that only writes to disk if the database is not connected, or has been paused. If the database is available, the data will be transmitted directly to the database. This means that there is no penalty for using store and forward during normal operation. The DataHub store and forward mechanism uses two levels of disk caching to ensure that all data gets logged, and nothing is lost.

When the database first becomes available after an outage, the DataHub starts writing cached values to the database, and continues writing new values to the cache. In this way, the values are inserted into the database in the order in which they are generated by the system. Once the cache is cleared, the DataHub then starts writing new values directly to the database.

Enable store and forward

Activates the store and forwarding feature.
Always write queue to disk
Data in the transaction queue will be written to disk cache first, and from there to the database. The safest protection against a crash is to check this box, and uncheck Delay writes to disk (below).

Never write queue to disk
The data in the transaction queue will be only stored in memory, and never written to disk.

Delay writes to disk
Data in the transaction queue will be written to disk at the most opportune times. The safest protection against a crash is to uncheck this box, and check Always write queue to disk (above).

Allow duplicates while forwarding stored data
If the network breaks while transmitting data from a cache, the Cogent DataHub needs to know how to handle any already-sent data when it reconnects. Leaving this box unchecked will require the Cogent DataHub to track its cache position at all times, and modify that information each time a value is sent. This will impact the speed of every transmission, but it will ensure that no values get transmitted twice.

Checking this box will cause the DataHub to simply start from the beginning of the queue or cache on each reconnect, and retransmit some data. This significantly reduces data-handling complexity and decreases transmission rates. This option is particularly useful if network breaks are frequent and some duplication of logged data is acceptable.

Show statistics in tray menu
Adds a Data Logging entry to the DataHub's system tray menu, which lets you open a statistics window:

Transactions sent successfully:
The number of transactions that were sent, either directly to the database, or to the disk cache.
Transactions / sec (10 sec window):
The sending rate for transactions, calculated over the past 10 seconds.

Transactions currently on queue:
The number of transactions in the queue.

Results currently on queue:
Not yet documented.

Transactions rejected (full queue)
The number of transactions that were rejected from the queue because it was full.

Transactions stored in L1 cache
The number of transactions taken off the queue and put into the first-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L1 cache
The number of transactions that were not able to be stored in the first-level cache.

Transactions stored in L2 cache
The number of transactions taken off the queue and put into the second-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L2 cache
The number of transactions that were not able to be stored in the second-level cache.

Transactions forwarded from cache
The total number of transactions forwarded from both caches. This number should be the sum of L1 and L2, once all transactions have been forwarded, and as long as the DataHub was started up with no cache on disk.

Transactions failed from cache
The number of transactions attempted from cache, could not be successfully delivered, and were stored for later transmission. This phenomenon may occur the first time that the DataHub learns that the database is not available. For example, you’ll see this for every network break if you’ve checked Always write queue to disk.

Cache directory:
The path and directory name for the cache.

Maximum cache size (MB):
The amount of disk space to allocate for the cache, in megabytes.

Setting up the DSN (Data Source Name)

A DSN is a Data Source Name. Windows uses this name to identify the database you want to connect to. This tab lets you select an existing DSN, or create a new one if necessary.
Open the ODBC Data Logging window

1. In the Cogent DataHub Properties window, select **Data Logging**.
2. In the **Configure Database (ODBC) Data Logging** section click the **Configure** button.

The **Data Logging Configuration** interface is explained in detail in the section called “Configuring the Queue, Store and Forward”.

3. Configure your DSN as explained below.

Selecting a DSN

1. To select a DSN, choose one from the drop-down box, and then enter the **user name and password**, if applicable.

   If your DSN is using Windows Authentication, leave the **User** and **Password** fields blank.

2. Click the **Connect** button. A "Connected to . . ." message should appear in the message box. If you get an error message in the box, consult your system administrator.
Creating or Configuring a DSN

Windows has different configuration programs for 32-bit and 64-bit DSNs. The Cogent DataHub uses 32-bit DSNs. To configure a 32-bit DSN in a 64-bit system, you need to run:

c:\windows\syswow64\odbcad32.exe

1. To create or configure a DSN, click the **Open DSN Administrator** button. This opens the ODBC Data Source Administrator window.

2. Select the **User DSN** or **System DSN** tab, depending on how you plan to access your database.
   
   A user DSN is only available to the current user account, while a system DSN is available to any user account on the computer.

3. Now you can add a new database or configure an existing one.

   **Add a new database**

   1. Click the **Add** button. The Create New Data Source window will open, displaying a list of data source drivers.

   2. Select the data source driver that corresponds to your ODBC database. A data source setup window will open. Each data source setup window is different, but you should be able to find the appropriate entry fields easily enough.

   3. Enter the data source name and select the database.

   4. Enter any other required or optional information such as login name, password, etc. What entries need to be made and where they are entered depends on the particular data source setup window you are using.

   5. Click **OK** to return to the ODBC Data Source Administrator window. You should be able to see the new database and driver listed. If you need to make any changes, you can configure an exiting database, as explained below.

   **Configure an existing database**

   1. Select a data source name and click the **Configure...** button. This takes you to the data source setup window (explained above) where you can make changes to the configuration.
2. Make your changes and click OK to return to the ODBC Data Source Administrator window. Any time you need to make a change, you can go to this window.

4. When you are satisfied everything is correct, click the OK button to exit the ODBC Data Source Administrator.

Once you have created a DSN, you can select it as explained above.

**Configuring a Database Table**

After you have set up a DSN you can create a table or select an existing one, and then assign points and properties to the columns of the table.

![Write to a Database (ODBC) interface](image)

**Table Selection or Creation**

You need to either select an existing table, or create a new one.

- **Selecting a table** allows you to use a table that you have created in your database program. When you select an existing table, you cannot add columns or change column names. To select a table, choose a table name from the drop-down list.

- **Creating a table** provides a way to design and create a table from within this interface.
  1. From the **Table Name** drop-down list, select **Create a new table**.
  2. In the dialog box, type in a table name and click OK.
     Now you will have to add at least one column.
  3. Click the **Add Column** button to create a column.
  4. In the dialog box, type in a column name and click OK.
     At this point you can add more columns, or you can assign points (as explained below). You can easily rename, insert, or delete a column by right-clicking on the col-
um name and selecting **Rename Column**, **Insert Column**, or **Delete Column** from the pop-up menu.

5. When all of your columns have been created and named, you can create the table in the database. Once created, you cannot add columns to the table. To create the table, click the **Create Table** button.

If, after creating the table, you need to make changes to it, and it has no data in it that you need, you can click the **Delete Table** button. This will delete the table from the database but will leave all of the configuration you’ve done intact in this window. You can then add more columns or make changes to points and properties and recreate the table. This can be done as often as you need to.

**Add New Rows or Modify the Same Row**

For any given table, whether existing or newly-created, you will need to decide whether you are going to add new rows, or modify the same row with new data each time it changes.

- **Add New Rows** creates a new row in the table each time data is logged.
- **Modify Same Row** writes to the same row in the table each time data is logged.

**Assigning a Key**

You have the option to make one column, often the first column in a table, a key column.

- **An auto-incrementing key** is commonly configured for **Add New Rows**. A key column is optional for adding new rows, but if you choose to have one, it **must** be auto-incre-atinct. Please refer to the section called “Key Columns” for more details. An auto-incrementing key can be configured as follows:

  **Item:** Choose `<key>` from the drop-down list.
  **Property:** (none)
  **Data Type:** The appropriate auto-incrementing integer type for your database.

- **A single key column** is required for **Modify Same Row**. The DataHub uses the key value to determine which row to modify. Please refer to the section called “Key Columns” for more details. The key can be configured as follows:

  **Item:** Type in a DataHub point name, or any other value.
  **Property:** `<key>`
  **Data Type:** The appropriate type for your database (typically VARCHAR, NVARCHAR or CHAR).
Due to a Windows bug, in the first column the interface won't respond unless you click directly below the text of the column name. We are working to resolve this.

Assigning points and properties

When you have selected a table, or you have at least one column in a table you are creating, you can assign points and their properties to the various columns.

**Item**

First, choose a point from the point-picker list on the right. Then click in the **Item** row and select `<point>` from the drop-down list. Optionally, you can type in the name of the point. Leaving the **Item** blank allows you to choose the **Property** of `clock` to display the system time, or `clockms` to display the number of milliseconds after the second of the system time.

**Property**

Select which property of the point you want written to the database in this column:

- **name**
  - The name of a the point shown in the **Item** field.
- **value**
  - The value of the point shown in the **Item** field.
- **quality**
  - The quality of the point shown in the **Item** field.
- **timestamp**
  - The time stamp of the point shown in the **Item** field. This will include the milliseconds, but many databases, such as MS Access, ignore the milliseconds and store only the seconds. Other databases such as MySQL and MS SQL Server include the milliseconds in a time stamp. For example:
  
  - Databases like MySQL and MS SQL Server:

<table>
<thead>
<tr>
<th></th>
<th>Column A</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enter for Property</strong></td>
<td><strong>timestamp</strong></td>
</tr>
<tr>
<td><strong>Enter for Data Type</strong></td>
<td><strong>datetime or timestamp or date</strong></td>
</tr>
</tbody>
</table>

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
Write to a Database

• Databases like MS Access:

<table>
<thead>
<tr>
<th>Enter for Property</th>
<th>Column A</th>
<th>Column B</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>timems</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enter for Data Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>datetime</td>
<td>number or integer</td>
</tr>
</tbody>
</table>

**timems**
The millisecond component of the `timestamp`, generally used in conjunction with `timestamp`. You only need this if your database cannot store the millisecond component of `timestamp`.

**timestampUTC**
The same as the `timestamp`, but in UTC time. You can use `timems` in conjunction with this as well.

**clock**
The current system time. This will include the milliseconds, but like `timestamp` (above) many databases ignore the milliseconds and store only the seconds.

**clockms**
The millisecond component of `clock`, generally used in conjunction with `clock`. Like `timems` (above), you only need this if your database cannot store the millisecond component of `clock`.

**Using clock and clockms** some tips:

• You must leave the Item field blank to select either of these options.

**Example** If the time is 12:34:56.789, `clock` will be written as 12:34:56.789 in databases that accept milliseconds, and as 12:34:56 in databases that do not. A `clockms` property will be written as 789 in all databases.

• The `clock` and `clockms` properties allow you to log the system time as a column in the table, so that your record can contain the system time along with a number of different point values, for example:

<table>
<thead>
<tr>
<th>System Time</th>
<th>Point1</th>
<th>Point2</th>
<th>Point3</th>
</tr>
</thead>
<tbody>
<tr>
<td>08:12:56.000</td>
<td>43.883</td>
<td>3.727</td>
<td>213.905</td>
</tr>
</tbody>
</table>

**Data Type**
The data type that the database should associate with the property.
Data Size
In some cases this is entered automatically, in other cases it is not used, but sometimes it is possible or necessary to enter a size, such as the number of characters in a text string, or the number of bytes.

Transform(x)
This allows you to modify the entry or to insert a text string. For example:
- \((x \times 100) + 25\) could be used to multiply a point value by 100 and add 25 to the result.
- "Tank Level" would insert the string Tank Level instead of, say, the point name. Any valid Gamma expression can be used.

Key Columns
A database table can have a special column (or set of columns) that is designated as a key. Every value in the key column(s) is unique. By designating a key in a table, you are providing the database engine with a guarantee that any search on the key will produce either zero or one row. Effectively, when a search matches a value in the key column, the row containing that key is guaranteed to be unique in the table.

A database table does not require a key. The key is effectively a hint to the database engine to improve efficiency and to guarantee uniqueness. You can still search and modify a table that has no key.

When working with the DataHub Data Logging interface, you have the option of specifying a key column. The interface will only allow single-column keys. You will not be able to specify more than one key column, and if you choose an existing table with multiple key columns, the interface will fail.

Your choice of key will depend on whether you have chosen Add New Rows or Modify Same Row for your database table.

Adding New Rows
If you choose to add (insert) new rows into your database table, you do not require a key column. Each new logging event will create a new row and populate it with the data that you have configured. Since you are continually adding rows to the table, there is no data that can be guaranteed to be unique within a column. Any one of point name, quality, value or timestamp may be repeated.

Consequently, if you do wish to have a key column in your table, it must have one important property: it must be automatically generated and guaranteed to be unique. The key value cannot be derived from a DataHub point. Most database engines support the concept of a counter or an auto-increment numeric value. If you choose to use a table containing a key, the key column must be of the appropriate auto-incrementing integer type for your database.

You can designate a key by choosing the <key> option for the Item entry of the database table when configuring a table through the Data Logging interface. If you are using an ex-
isting table that already contains a key, you must specify in the Data Logging interface that the column is a key column.

**Modifying the Same Row**

If you choose to modify a row in the database, such that any new data will overwrite the existing data in the row, you must be able to uniquely identify that row. This means that you **must** have a key column.

The key column can be any type, and does not need to be auto-incrementing. Since the row is overwritten whenever new data is available, no new key value is generated. It is common to supply the DataHub point name as a key, with the key field defined as a string type (typically VARCHAR, NVARCHAR or CHAR). You can do this by selecting a point in the point picker tree, then choosing the `<point>`; option for the **Item** entry of the database column, and `<key>` for the **Property** field.

If you wish to specify a key value other than a point name, type your own value into the **Item** entry instead of using the point name. You must then choose `<key>` for the **Property** entry.

The key value you enter here is evaluated in Gamma as an expression or symbol. If it is a string, it must be written inside quotes to prevent causing an undefined symbol error. Also, if you are using the **Remote Config** interface, the expression will appear in the DataHub's point list.

**Assigning a Trigger**

A trigger is an event that causes a row of data to be written to your database table. A trigger event can be either a point value change, a timer event, or a calendar event. You can assign a different trigger for each row, or an identical trigger to any number of rows. An action can be configured to execute on every trigger event, or you can assign **trigger conditions** that are evaluated whenever a trigger occurs, to determine if the action should be executed.

The three kinds of triggers are:
• **Point Change** fires whenever a specified trigger point changes.

  1. Type the name of the point into the **Point Name** box, or select the point using the data tree on the right, then click the + button.

  2. (Optional) Enter a value deadband if you want to filter out extraneous data. The number you enter will specify a high and low (plus or minus) range. Any value change falling within that range will not cause the trigger to fire. A positive or negative change greater than this value will activate the trigger and cause the row to be written.

    To create a trigger that gets reset automatically, please refer to An Auto-Resetting Trigger in the section called “Setting Trigger Conditions”.

• **Repeat Timer** fires cyclically, each time the number of seconds elapses.

• **Time of Day** fires at the time you specify. You can enter:

  • A number, indicating a specific value. For example, a 0 in the seconds field would cause the event only on the 0th second of the minute. A 30 would indicate only on the 30th second of the minute.

  • A list of numbers, separated by commas. For example, entering 0,15,30,45 in the minutes field would indicate that the event should fire on the hour and at 15, 30 and 45 minutes past the hour.

  • A range of numbers, separated by a dash. For example, entering 8–18 in the hours field would indicate that the event should fire every hour from 8 a.m. to 6 p.m.

    Ranges can be mingled with lists, as in 0,4,8–16,20.

  • An asterisk (*) indicates that the event should fire for every possible value of the field. For example, a * in the seconds field would cause the event to fire every second. A * in the hours field would cause the event to fire every hour.

    To regularly log a record on specific days of the week, please refer to the section called “Setting Trigger Conditions”.

The ranges of the fields are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>1970–*</td>
</tr>
<tr>
<td>Month</td>
<td>1–12</td>
</tr>
<tr>
<td>Day</td>
<td>1–31</td>
</tr>
<tr>
<td>Hour</td>
<td>0–23</td>
</tr>
<tr>
<td>Minute</td>
<td>0–59</td>
</tr>
<tr>
<td>Second</td>
<td>0–59</td>
</tr>
</tbody>
</table>

The year and month are entered differently here than for the Gamma lo-caltime function, as explained in Time Conditions.

Examples:

• These entries:
would cause a row to be logged at 8:45 every day, every month, and every year.

• These entries:

would cause a row to be logged every hour on the 15th day of each month, every year.

• These entries:

would cause the event to fire every second for 5 minutes, every two hours between 8 a.m. and 6 p.m.

• After fires once, when the specified number of seconds elapses.
• None configures no trigger.

**Setting Trigger Conditions**

Each logging action can have up to four conditions that determine whether a row gets written to the database when the trigger fires.
Fill in the conditions according to the guidelines below. Check the box next to the condition to apply it. As you make entries, the corresponding Gamma code will appear in the display. Gamma is the DataHub’s built-in scripting language. The code that appears in the Expression box is the actual code that gets run by Gamma. The order of precedence for “And” and “Or” operators (&& and ||) is first And, then Or.

Point Value Conditions

Point names can be entered on either or both sides of the comparison. They can be picked from the data tree list, or typed in. Each point name needs to have a dollar sign ($) in front of it to indicate to Gamma that this is a DataHub point. You can put numerical values into either side of the comparison.

When you enter a point name in a condition field, the current value of the point will be used in the evaluation. For example, you could define a condition that states that whenever the trigger event occurs, the action will only be executed if another point value is within a certain range.

There are some automatic variables available for working with point values:

- lasttrigger - the value of the trigger point the last time this trigger was fired (even if the condition failed).
- thistrigger - the value of the trigger point now (even if the condition failed).
- lastevent - the value of the trigger the last time the event was actually executed (the condition succeeded).
- this - the trigger point itself, not the value of the point.

You can use these variables in any condition that is triggered by a data value change. For example, you could create some conditions like this:
Time Conditions

This provides an additional way to restrict the time, day, month, etc. when a message gets sent. In addition to the options on the triggers, here you have day-of-week condition statements which can give you more flexibility for events based on specific days of the week. These will work with any type of trigger event.

You can use the Gamma functions `clock` and `localtime` to specify particular days of the week. For example, these entries:

```plaintext
(localtime(clock()).wday > 0 && localtime(clock()).wday < 6)
```

would create this Gamma code:

which would cause data to be logged only Monday through Friday. The function `localtime` returns a class whose members contain information about the date, as follows:

- `.sec` The number of seconds after the minute (0 - 59).
- `.min` The number of minutes after the hour (0 - 59).
- `.hour` The number of hours past midnight (0 - 23).
- `.mday` The day of the month (1 - 31).
- `.mon` The number of months since January (0 - 11)
- `.year` The number of years since 1900.
- `.wday` The number of days since Sunday (0 - 6).
- `.yday` The number of days since January 1 (0 - 365)
- `.isdst` 1 if daylight saving time is in effect, 0 if not, and a negative number if the information is not available.

The year and month are entered differently here than for Time of Day trigger conditions, as explained in the section called “Assigning a Trigger”.

[Diagram of Time Conditions]
There are two automatic variables available for working with time values:

- `lasteventtime` - the system clock time (UTC floating point) that the last event was executed.
- `curtime` - the system clock time (UTC floating point) when the current trigger occurred.

**Custom Conditions**

If the conditions you need to meet are beyond the scope of this interface, you can use a Gamma function to express virtually any condition you need. Then you can insert the function into one of the condition boxes, and set a condition based on the return value of the function.

To do this you can create a DataHub script (.g file) that contains only the functions you will be using for conditions, without any classes or methods. For example, here is the complete contents of such a file, named `MyConditions.g`:

```g
function MyFunction ()
{
    myvalue = $DataSim:Sine;
    princ("Value when the trigger fired: ", myvalue, "\n");
    myvalue;
}
```

This function prints the value of the `DataSim:Sine` point, and returns its value. We can use this function as a condition by calling it from one of the condition boxes in the interface, like this:

When the trigger fires, `MyFunction` is called, and the return value gets checked to see if it is less than 0.4. If so, the data gets logged.

Below are two practical examples demonstrating custom functions. The first creates an automatically resetting trigger, and the second lets you test for changed values before logging a point, which is useful if you are logging based on a timer.

**An Auto-Resetting Trigger**

This script can turn any DataHub point into a trigger that automatically resets. To use it, you first need to load and run the `TriggerFunctions.g` script (shown below). Then, if you put this formula:

```
HighWithReset($TriggerPoint) != nil
```

into the condition boxes, whenever the `TriggerPoint` changes to a non-zero number in the DataHub, your trigger will fire. The script waits for a millisecond, then resets
the TriggerPoint back to zero. The second function works similarly, but triggers on a change to zero, instead of a change to a non-zero number.

**TriggerFunctions.g**

```g
test a trigger point for a non-zero value. If the point is non-zero, create a delayed event to reset the point to zero, and return true, indicating that the condition has succeeded and the action should proceed. If the value is 0, then simply return nil indicating that the action should not proceed. We need to test for zero because when we reset the trigger point to zero a second data change event will occur.

The argument is unevaluated, so the condition should look like this:

```
    HighWithReset($default:TriggerPoint) != nil
```
```

function HighWithReset(!triggerpoint)
{
    local value;
    if (!undefined_p(value = eval(triggerpoint)) && value != 0)
    {
        after(0.001, `setq(@triggerpoint, 0));
        t;
    }
    else
    {
        nil;
    }
}
```

/** This is the inverse of HighWithReset (see above). If the trigger point is zero, perform the action and set the trigger point to 1. If the trigger point is non-zero do nothing and return nil. */

function LowWithSet(!triggerpoint)
{
```
```
local value;
if (!undefined_p(value = eval(triggerpoint)) && value == 0)
{
    after(0.001, `setq(@triggerpoint, 1));
    t;
}
else
{
    nil;
}
}

Test for Changed Values Before Logging

If you are logging data based on a timer, you can use this script to check the previously
logged value, and only log a new row in the database if the value has changed. To use the
script, you first need to load and run the LogFunctions.g script (shown below and in-
cluded in the installation archive). Then, if you put this formula:

HasValueChanged(#$TestPoint) != nil

into the condition boxes, when the TestPoint has not changed, the data for that row will
not be logged.

The call to HasValueChanged uses the syntax #$pointname to specify the point. If you
select the point from the tree to the right and press the + button, it will not include the #
symbol. You need to manually add it. You can see the complete condition expression in
the Expression: line beneath the condition selectors, like this:

LogFunctions.g

/*
 * Functions that can be used by the Condition section of a Data
 * Logging configuration to filter whether data is actually written
 * to the database when a trigger occurs.
 */
//// Change this to nil to silence the debugging statements that will
//// be written to the script log.
LogFunctionDebug = t;

/*
 * Track the previous value of a DataHub point and only allow the
 * logging action if the DataHub point value has changed since the
 * last time we logged it.
 *
 * Takes a single argument that is either a point symbol or a point
 * name as a string.
 *    e.g., "DataPid:PID1.Mv"
 *          #$DataPid:PID1.Mv
 *
 * Notice that a point symbol must be specified with the # operator
 * before the symbol to ensure that the symbol is passed, not the
 * value of the point.
 */

function HasValueChanged(point_symbol)
{
    local ptsym = symbol(point_symbol);
    local oldvalue = getprop(ptsym, #last_log_value);
    local curvalue;
    local result = nil;

    if (!undefined_p(curvalue = eval(ptsym)))
    {
        if (LogFunctionDebug)
            princ("Checking value of ", ptsym, ": previous = ", oldvalue,
                  ", current = ", curvalue, "\n");
        if (oldvalue != curvalue)
            result = t;
        setprop(ptsym, #last_log_value, curvalue);
    }
    else
    {
        if (LogFunctionDebug)
            princ("Log condition: value of ", ptsym, " is undefined\n");
    }
    result;
}
Configured Actions

A configured action will cause a row of data to be written into the table specified in your DSN, based on a trigger and optional conditions. It is the end result of your configuration activities in this interface. The Configured Actions list shows the actions you have configured, and allows you to create, modify, or remove actions, as well as turn them on or off.

The list of configured actions shows the actions you have already configured. Selecting an existing action from the list automatically fills in the DSN, Table, Trigger, and Condition tabs with its information. Checking or unchecking the On box at the left lets you switch the action on or off.

The Create button creates an action for the information currently entered in the DSN, Table, Trigger, and Condition tabs. If you press the Create button while a configured action is selected, it creates a duplicate of that configured action and adds it to the list. This is a quick way to configure similar actions.

The Modify button overwrites the selected configured action with the information currently entered in the DSN, Table, Trigger, and Condition tabs.

The Remove button removes a configured action.

Once a configured action has been created or modified, the changes won't take effect until you click the Apply or Done button. Each write action to the database gets logged in the DataHub Script Log. This allows you to check for error messages and ensure that your data is being written successfully. You can also verify the writes by querying the database itself.
Query a Database

Introduction

The Cogent DataHub can query any ODBC compliant database, and bring the data back as DataHub points, or into a single point in XML format for use in $wvp;.

The Cogent DataHub’s ODBC Data Logging interface provides an easy way to connect to a database and submit a query. There are two options for writing the query to the DataHub: one DataHub point per row, or the whole query results in a single DataHub point. The fastest way to learn how to use the interface is by using the Quick Start.

Quick Start

Here’s how to configure the Cogent DataHub to query a database of your choice.

Click here to watch a video.

Open the Query ODBC Database window

1. In the Cogent DataHub Properties window, select Database.
2. In the Query a Database (ODBC) section, click the Configure button.

This opens the Query a Database (ODBC) window, shown below.

Connect to the Database

1. Select the 1. DSN tab. A DSN is a Data Source Name. Windows uses this name to identify the database you want to connect to.
2. From the drop-down box, select a DSN. If you do not have any DSNs, or you wish to create a new DSN, you can do this by opening the DSN Administrator. Please refer to Setting up a DSN for more details.

3. Enter the user name and password (if required), and click the Connect button. A "Connected to ..." message should appear in the message box. If you get an error message in the box, consult your system administrator.

Configure a Query

For this and other queries in this chapter, we will use a simple example database named **test** with a table named **querytest** that has 4 columns and 3 rows.

<table>
<thead>
<tr>
<th>ID</th>
<th>VariableName</th>
<th>VariableValue</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Testpoint1</td>
<td>95</td>
<td>First test point</td>
</tr>
<tr>
<td>2</td>
<td>Testpoint2</td>
<td>87</td>
<td>Second test point</td>
</tr>
<tr>
<td>3</td>
<td>Testpoint3</td>
<td>19</td>
<td>Third test point</td>
</tr>
</tbody>
</table>

In our example, we will be writing the contents of the **VariableName** and **VariableValue** columns to the DataHub.

1. Select the **2. Query** tab.
2. In the **Label:** field, enter a name for this query, such as **TestQuery**. This can be any string.

3. Enter a valid SQL query. As an example, for our small database, we can use a simple SQL query:

   ```sql
   SELECT * FROM database.table LIMIT 3;
   ```

   If you use a `SELECT * FROM` query like this on a large database with a frequent timer, we recommend limiting the number of rows returned, to prevent hanging the database.

4. Press the **Submit** button to submit your query to the database for a check. If the query is not valid, a message will pop up informing you about any errors. You can also open the **Script Log** to see more information about your connection to the database, and the results of your query.

5. There are two options for how a query is written to the DataHub. Here we will introduce the **One point per row** option. For information about the **Whole data set in a point** option, please refer to the section called “Configuring a Database Query”.

   Select **One point per row**.
6. In the **Column Name / Point Name** dropdown list, choose the column name in your database that contains the point names that you will be using. For instance, in our demo database that column is named **VariableName**.

7. In the **Column Name / Value** dropdown list, choose the column name in your database that contains the values for the point names that you will be using. For instance, in our demo database that column is named **VariableValue**.

8. In the **Replace domain** field, enter the name of a new or existing domain. We've put in a name for a new domain, **TestDomain**.

9. Click the **Create** button.

A new configured action should appear in the list. For more information about configured actions, please refer to the section called “Configured Actions”.

Next, to get the DataHub to make the query, you need to assign a trigger.

**Assign a Trigger**

For this example, we will set a repeat timer to re-query the database every 5 seconds.

1. Select the **3. Trigger** tab.
2. Select **Repeat Timer** and enter **5**.

3. In the **Configured Actions** box, make sure that the configured action you just created is highlighted. If not, click on it to highlight it. Then click the **Modify** button.
Your configured action should now display **every 5 sec** in the **Trigger** column.

4. Click the **Apply** button to activate the configured action.

5. Open the Data Browser window. After 5 seconds, the points should appear in the Data Browser:

![Data Browser](image)

To test the updates, you can change a value associated with a point name in the database, and the change should appear in the Data Browser within 5 seconds.

Should you not see the data points, or changes made to the values, you can open the **Script Log** to check for error messages and ensure that your query is being sent every 5 seconds.

If all is working as described, you have configured an action that queries a database for and writes the results into the DataHub. The remaining sections in this chapter explain the interface in more detail, and introduce the option of setting specific **conditions** for queries, if desired.

### Setting up the DSN (Data Source Name)

A **DSN** is a Data Source Name. Windows uses this name to identify the database you want to connect to. This tab lets you select an existing DSN, or create a new one if necessary.

#### Open the Query a Database (ODBC) window

1. In the Cogent DataHub Properties window, select **Database**.
2. In the **Query a Database (ODBC)** section click the **Configure** button.
3. Configure your DSN as explained below.

![Configure ODBC Database Query](image)

**Selecting a DSN**

1. To select a DSN, choose one from the drop-down box, and then enter the *user name and password* (if required).

   If your DSN is using Windows Authentication, leave the *User* and *Password* fields blank.

2. Click the *Connect* button. A "Connected to ..." message should appear in the message box. If you get an error message in the box, consult your system administrator.

**Creating or Configuring a DSN**

Windows has different configuration programs for 32-bit and 64-bit DSNs. The Cogent DataHub uses 32-bit DSNs. To configure a 32-bit DSN in a 64-bit system, you need to run:

```
c:\windows\syswow64\odbcad32.exe
```

1. To create or configure a DSN, click the *Open DSN Administrator* button. This opens the ODBC Data Source Administrator window.
2. Select the **User DSN** or **System DSN** tab, depending on how you plan to access your database.

   A user DSN is only available to the current user account, while a system DSN is available to any user account on the computer.

3. Now you can add a new database or configure an existing one.

   **Add a new database**
   1. Click the **Add** button. The Create New Data Source window will open, displaying a list of data source drivers.
   2. Select the data source driver that corresponds to your ODBC database. A data source setup window will open. Each data source setup window is different, but you should be able to find the appropriate entry fields easily enough.
   3. Enter the data source name and select the database.
   4. Enter any other required or optional information such as login name, password, etc. What entries need to be made and where they are entered depends on the particular data source setup window you are using.
   5. Click **OK** to return to the ODBC Data Source Administrator window. You should be able to see the new database and driver listed. If you need to make any changes, you can configure an exiting database, as explained below.

   **Configure an existing database**
   1. Select a data source name and click the **Configure...** button. This takes you to the data source setup window (explained above) where you can make changes to the configuration.
   2. Make your changes and click **OK** to return to the ODBC Data Source Administrator window. Any time you need to make a change, you can go to this window.

   Once you have created a DSN, you can select it as explained above.

   **Configuring a Database Query**

   After you have set up a DSN you can create a database query.
Preparing the Query

For this and other queries in this chapter, we will use a simple example database named test with a table named querytest that has 4 columns and 3 rows.

- In the Label: field, enter a name for this query, such as TestQuery. This can be any string.
- Enter a valid SQL query.
- The Submit submits your query to the database for a check. If the query is not valid, a message will pop up informing you about any errors. You can also open the Script Log to see more information about your connection to the database, and the results of your query.

Write One Point Per Row to the DataHub

- The One point per row option lets you assign one point in the DataHub for each
uniquely named item in one column of a database.

- The drop-down Column Name lists allow you to choose the columns that contain the point name and its value that will appear in the DataHub, as well as an associated time stamp and quality.
- The Transform(x) fields let you to perform transformations on the data in each row of the corresponding column. These transformations are written in Gamma, the DataHub Scripting language.

For example, an entry of `string("MyTag", x)` under the Point Name would change points labelled Pump1 and Pump2 to MyTagPump1 and MyTagPump2. Or, an entry of `10 * x` under the Value would multiply each value by 10.

- The output of One point per row for our example database would appear in the DataHub Data Browser like this:

**Put a Whole Data Set into One DataHub Point**

The Whole data set in point option lets you put the entire results of the query into one DataHub point.

The results can be written in XML format (the default), or for Excel by selecting the Use Excel format instead of XML option.

For example, using the XML option, running our query of `SELECT * FROM test.query-test LIMIT 3;` on our example database would write the following XML-formatted data as the value of a single point:

```xml
<TableData>
  <column name="ID" type="int" datatype="-6" />
</TableData>
```
This data can be parsed by an XML parser, and is particularly useful for displaying in DataHub WebView, using the WebView Filtered Data Table control. In DataHub WebView, you would add a Filtered Data Table to a page, and configure this point as the **Items Source**:

![Filtered Data Table Control](image)
Query a Database

The Excel output generated with the **Use Excel format instead of XML** option would look like this when dragged and dropped into an Excel worksheet:

![Excel output](image)

### Dynamic Database Queries

It is possible to create dynamic database queries by inserting Gamma expressions into the query. The value, time, and quality attributes of the DataHub points are accessed by using the following syntax:

<table>
<thead>
<tr>
<th>Button</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td><code>domainname:pointname</code></td>
<td><code>DataSim:Sine</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>&lt;%= $domainname:pointname %&gt;</code></td>
<td><code>&lt;%= $DataSim:Sine %&gt;</code></td>
</tr>
<tr>
<td>Time</td>
<td><code>&lt;%= PointTimeString (#$domainname:pointname) %&gt;</code></td>
<td><code>&lt;%= PointTimeString (#$DataSim:Sine) %&gt;</code></td>
</tr>
<tr>
<td>Quality</td>
<td><code>&lt;%= PointQualityString (#$domainname:pointname) %&gt;</code></td>
<td><code>&lt;%= PointQualityString (#$DataSim:Sine) %&gt;</code></td>
</tr>
</tbody>
</table>

In this syntax, the special characters are used as follows:

<table>
<thead>
<tr>
<th>Character</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;% ... %&gt;</code></td>
<td>The enclosed expression will be evaluated by Gamma, the DataHub scripting language.</td>
</tr>
<tr>
<td>$</td>
<td>Indicates to Gamma that this is a DataHub point name.</td>
</tr>
<tr>
<td>PointTimeString()</td>
<td>A Gamma function that returns the timestamp of a DataHub point in an easily readable format.</td>
</tr>
<tr>
<td>PointQualityString()</td>
<td>A Gamma function that returns the quality of a DataHub point, as a text string.</td>
</tr>
</tbody>
</table>
**Character** | **Use**
--- | ---
# | Protects the DataHub point from being evaluated by Gamma until the function is called.

The query string is processed using the "asp" Gamma processor, so you can insert any Gamma string into the query.

**Example**

A query using a selection criteria based on the point PLC1.ResourceX in data domain records would look something like this:

```sql
select top 1 OrderNo, from.Orders
where Resource=<%= $records:PLC1.ResourceX %>
and Priority=10 order by DueDate asc
```

**Assigning a Trigger**

A trigger is an event that causes your query to be sent to the database. A trigger event can be either a point value change, a repeat timer, a time of day timer, or an "after" timer. You can assign a different trigger for each query, or an identical trigger to any number of queries. A query action can be configured to execute on every trigger event, or you can assign trigger conditions that are evaluated whenever a trigger occurs, to determine if the action should be executed.

The four kinds of triggers are:

- **Point Change** fires whenever a specified trigger point changes.
  1. Type the name of the point into the **Point Name** box, or select the point using the data tree on the right, then click the + button.
  2. (Optional) Enter a value deadband if you want to filter out extraneous data. The number you enter will specify a high and low (plus or minus) range. Any value
change falling within that range will not cause the trigger to fire. A positive or negative change greater than this value will activate the trigger and cause the row to be written.

To create a trigger that gets reset automatically, please refer to An Auto-Resetting Trigger in the section called “Setting Trigger Conditions”.

- **Repeat Timer** fires cyclically, each time the specified number of seconds elapses.
- **Time of Day** fires at the time you specify. You can enter:
  - A number, indicating a specific value. For example, a 0 in the seconds field would cause the event only on the 0th second of the minute. A 30 would indicate only on the 30th second of the minute.
  - A list of numbers, separated by commas. For example, entering 0, 15, 30, 45 in the minutes field would indicate that the event should fire on the hour and at 15, 30 and 45 minutes past the hour.
  - A range of numbers, separated by a dash. For example, entering 8–18 in the hours field would indicate that the event should fire every hour from 8 a.m. to 6 p.m.
  - An asterisk (*) indicates that the event should fire for every possible value of the field. For example, a * in the seconds field would cause the event to fire every second. A * in the hours field would cause the event to fire every hour.

To regularly log a record on specific days of the week, please refer to the section called “Setting Trigger Conditions”.

The ranges of the fields are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>1970–*</td>
</tr>
<tr>
<td>Month</td>
<td>1–12</td>
</tr>
<tr>
<td>Day</td>
<td>1–31</td>
</tr>
<tr>
<td>Hour</td>
<td>0–23</td>
</tr>
<tr>
<td>Minute</td>
<td>0–59</td>
</tr>
<tr>
<td>Second</td>
<td>0–59</td>
</tr>
</tbody>
</table>

The year and month are entered differently here than for the Gamma localtime function, as explained in Time Conditions.

Examples:
- These entries:
would trigger a query at 8:45 every day, every month, and every year.

- These entries:

```
Year: *  Month: *  Day: 15
```

would trigger a query every hour on the 15th day of each month, every year.

- These entries:

```
Year:   Month:   Day: 15
```

would trigger a query every second for 5 minutes, every two hours between 8 a.m. and 6 p.m.

- **After** fires once, when the specified number of seconds elapses.
- **None** configures no trigger.

### Setting Trigger Conditions

Each query can have up to four conditions that determine whether it gets sent to the database when the trigger fires.

Fill in the conditions according to the guidelines below. Check the box next to the condition to apply it. As you make entries, the corresponding **Gamma** code will appear in the display. Gamma is the DataHub’s built-in scripting language. The code that appears in the
**Expression** box is the actual code that gets run by Gamma. The order of precedence for "And" and "Or" operators (& & and ||) is first And, then Or.

**Point Value Conditions**

Point names can be entered on either or both sides of the comparison. They can be picked from the data tree list, or typed in. Each point name needs to have a dollar sign ($) in front of it to indicate to Gamma that this is a DataHub point. You can put numerical values into either side of the comparison.

When you enter a point name in a condition field, the current value of the point will be used in the evaluation. For example, you could define a condition that states that whenever the trigger event occurs, the action will only be executed if another point value is within a certain range.

There are some automatic variables available for working with point values:

- lasttrigger - the value of the trigger point the last time this trigger was fired (even if the condition failed).
- thistrigger - the value of the trigger point now (even if the condition failed).
- lastevent - the value of the trigger the last time the event was actually executed (the condition succeeded).
- this - the trigger point itself, not the value of the point.

You can use these variables in any condition that is triggered by a data value change. For example, you could create some conditions like this:

**Time Conditions**

This provides an additional way to restrict the time, day, month, etc. when a query gets sent. In addition to the options on the triggers, here you have day-of-week condition statements which can give you more flexibility for events based on specific days of the week. These will work with any type of trigger event.

You can use the Gamma functions `clock` and `localtime` to specify particular days of the week. For example, these entries:
would create this Gamma code:

```
(localtime(clock()).wday > 0 && localtime(clock()).wday < 6)
```

which would cause a query to be sent only Monday through Friday. The function `localtime` returns a class whose members contain information about the date, as follows:

- `.sec` The number of seconds after the minute (0 - 59).
- `.min` The number of minutes after the hour (0 - 59).
- `.hour` The number of hours past midnight (0 - 23).
- `.mday` The day of the month (1 - 31).
- `.mon` The number of months since January (0 - 11).
- `.year` The number of years since 1900.
- `.wday` The number of days since Sunday (0 - 6).
- `.yday` The number of days since January 1 (0 - 365)
- `.isdst` 1 if daylight saving time is in effect, 0 if not, and a negative number if the information is not available.

The year and month are entered differently here than for Time of Day trigger conditions, as explained in the section called “Assigning a Trigger”.

There are two automatic variables available for working with time values:

- `lasteventtime` - the system clock time (UTC floating point) that the last event was executed.
- `curtime` - the system clock time (UTC floating point) when the current trigger occurred.

**Custom Conditions**

If the conditions you need to meet are beyond the scope of this interface, you can use a Gamma function to express virtually any condition you need. Then you can insert the function into one of the condition boxes, and set a condition based on the return value of the function.

To do this is you can create a DataHub script (.g file) that contains only the functions you will be using for conditions, without any classes or methods. For example, here is the complete contents of such a file, named `MyConditions.g`:

```
function MyFunction ()
```
This function prints the value of the `DataSim:Sine` point, and returns its value. We can use this function as a condition by calling it from one of the condition boxes in the interface, like this:

```
{ myvalue = $DataSim:Sine;
  princ("Value when the trigger fired: ", myvalue, "\n");
  myvalue;
}
```

When the trigger fires, `MyFunction` is called, and the return value gets checked to see if it is less than 0.4. If so, the data gets logged.

**Example: An Auto-Resetting Trigger**

This script can turn any DataHub point into a trigger that automatically resets. To use it, you first need to load and run the `TriggerFunctions.g` script (shown below and included in the installation archive). Then, if you put this formula:

```
HighWithReset($TriggerPoint) != nil
```

into the condition boxes, whenever the `TriggerPoint` changes to a non-zero number in the DataHub, your trigger will fire. The script waits for a millisecond, then resets the `TriggerPoint` back to zero. The second function works similarly, but triggers on a change to zero, instead of a change to a non-zero number.

**TriggerFunctions.g**

```/*
* This file contains handy functions to perform more complex
* condition handling in the Condition tab of the data logging
* and email interfaces.
*/

/*
* Test a trigger point for a non-zero value. If the point is
* non-zero, create a delayed event to reset the point to zero,
* and return true, indicating that the condition has succeeded
* and the action should proceed. If the value is 0, then simply
* return nil indicating that the action should not proceed. We
* need to test for zero because when we reset the trigger point
* to zero a second data change event will occur.
* */```
* The argument is unevaluated, so the condition should look like this:
* HighWithReset($default:TriggerPoint) != nil
*/

function HighWithReset(!triggerpoint)
{
    local value;
    if (!undefined_p(value = eval(triggerpoint)) && value != 0)
    {
        after(0.001, `setq(@triggerpoint, 0));
        t;
    } else {
        nil;
    }
}

/*
* This is the inverse of HighWithReset (see above). If the trigger point is zero, perform the action and set the trigger point to 1. If the trigger point is non-zero do nothing and return nil.
*/
function LowWithSet(!triggerpoint)
{
    local value;
    if (!undefined_p(value = eval(triggerpoint)) && value == 0)
    {
        after(0.001, `setq(@triggerpoint, 1));
        t;
    } else {
        nil;
    }
}

Configured Actions

A configured action will cause your query to be sent to the database, according to the trigger and any conditions you have specified. It is the end result of your configuration activities in this interface. The Configured Actions list shows the actions you have configured, and allows you to create, modify, or remove actions, as well as turn them on or off.
The list of configured actions shows the actions you have already configured. Selecting an existing action from the list automatically fills in the DSN, Query, Trigger, and Condition tabs with its information. Checking or unchecking the On box at the left lets you switch the action on or off.

The Create button creates an action for the information currently entered in the DSN, Query, Trigger, and Condition tabs. If you press the Create button while a configured action is selected, it creates a duplicate of that configured action and adds it to the list. This is a quick way to configure similar actions.

The Modify button overwrites the selected configured action with the information currently entered in the DSN, Query, Trigger, and Condition tabs.

The Remove button removes a configured action.

Once a configured action has been created or modified, the changes won't take effect until you click the Apply or Done button. Each query to the database gets logged in the DataHub Script Log. This allows you to check for error messages and ensure that your query was sent successfully. You can verify the results of the query in the Data Browser.
Using the Web Server

Introduction

The Cogent DataHub has a built-in web server that lets you display live data in a web page in several different ways, depending on your needs. The data can come from any data source that is connected to the DataHub. A two-way data flow allows the user to view data and also write data back to the DataHub.

There are several different display technologies available:

- **ASP (Active Server Pages)** DataHub scripts embedded in an HTML page are run by the DataHub each time the page is requested by the client. The DataHub scripts generate the page content dynamically before sending the page to the browser. This is also called "server-side scripting".

- **AJAX (Asynchronous Javascript and XML)** Javascript code embedded in an HTML page is interpreted by the browser. The data updates automatically on the page, without the need for refresh. This is also called "client-side scripting". The DataHub supports polling AJAX, in which the Javascript code periodically sends commands to the DataHub to update the data on the browser page.

- **WebSocket** Not yet documented

Here is a comparison chart for these technologies:

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web browser support</strong></td>
<td>Desktop and mobile</td>
<td>Desktop and mobile</td>
<td>Desktop and mobile</td>
</tr>
<tr>
<td><strong>Plug-in/Active X required</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Update speeds</strong></td>
<td>No updates - Manual refresh required</td>
<td>Fast updates</td>
<td>Fast updates</td>
</tr>
</tbody>
</table>
### System requirements (CPU and memory)

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Moderate</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

### Bandwidth requirements (Will depend on point count and update rate)

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very low</td>
<td>Relatively high</td>
<td>Moderate to low</td>
<td></td>
</tr>
</tbody>
</table>

### Security (Password / SSL protection)

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Firewall friendly

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Licensing (Licenses required in addition to the standard DataHub Node license)

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataHub Web Server license</td>
<td>DataHub Web Server license</td>
<td>DataHub Web Server license, + TCP Link license for each connection</td>
<td></td>
</tr>
</tbody>
</table>

### Programming Requirements

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses DataHub scripting language</td>
<td>Uses JavaScript</td>
<td>Uses HTML or JavaScript</td>
<td></td>
</tr>
</tbody>
</table>

### Types of Application

<table>
<thead>
<tr>
<th></th>
<th>ASP</th>
<th>AJAX</th>
<th>WebSocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common uses</td>
<td>Good for displaying static or slow moving data. Used for shift reports and statistics.</td>
<td>Good for displaying fast moving data and alarm conditions. Used in web monitoring and trouble shooting applications.</td>
<td>Not yet documented.</td>
</tr>
</tbody>
</table>

These different display technologies can be used together in the same page. For example, we often use ASP code to dynamically create AJAX tables that display live data in a web browser. The ASP code does the repetitive task of writing table entries for each point in a specific data domain and the web browser interprets the resulting JavaScript and builds the AJAX display accordingly. You can also use ASP to access data from an ODBC database and display it as part of the web page, along with the live data from the DataHub.

**More about ASP**

- Most process data does not update very quickly, so ASP is suitable for a wide range of applications.
- ASP will typically use very few system resources and bandwidth, so it is good for low speed connections.
- ASP is the most efficient method for handling a large number of user connections.
- ASP pages are generated by a script running in the DataHub. This means you can use the script to bring in data from other sources, such as from SQL databases. The web page can then display data from both live sources and database or text file archives.
- Web pages that are generated by ASP scripts can usually be displayed on all desktop
and mobile web browsers. This is because while the scripts themselves are written in Gamma (the DataHub script language), the web pages they generate are delivered as plain HTML.

- In order to see new point values in an ASP page, you need to refresh the web page manually.

More about AJAX

- AJAX automatically updates the web page whenever the data changes in the DataHub, no page refresh is required.
- AJAX can handle high speed updates to a large number of users. However, system resources on the server will increase as the number of points, the number of users and the speed of the updates increases.
- AJAX displays are created using JavaScript programming, which means they are popular with web developers who wish to control all aspects of how the data is displayed in the web page.
- AJAX can be displayed using most modern desktop and mobile web browsers.
- AJAX web pages make requests for new data to the DataHub on a polling cycle. Fast polling rates and large data requests mean higher loads on the CPU.

More about WebSocket

- Not yet documented.
- Not yet documented.
- Not yet documented.

Configuring the DataHub Web Server

To configure the DataHub Web Server, follow these steps:

1. With the DataHub running, right click on the DataHub system-tray icon and choose Properties.
2. In the Properties window, select Web Server.
3. Check the Act as web server box.
4. The DataHub Web Server is preconfigured to run on port number 80, but you might need to change that setting in the Base Configuration section:
When using the DataHub Web Server to support WebSocket connections, we recommend using SSL.

When using SSL, we recommend using port 443.

Windows allows multiple users on a single TCP port, and never refuses a connection. However, this can cause irregular behavior. It is essential that the DataHub Web Server be the exclusive user of a port.

To get a list of which ports are in use on your machine, follow these steps:

a. From the Windows Start menu, choose Run.

b. Enter the executable name cmd.exe and click OK.

c. At the command prompt, type:

   ```bash
   netstat -p tcp -b -a -n
   ```

   The result is a table showing the tcp protocol and executable name of all programs in use. There are two columns of interest: Local Address and State. In
Local Address, the numbers at the end (after the colon) are the port number that the process is using. The State column shows the state of that process. The only state we are interested in is LISTENING. Whatever port you are using for the DataHub Web Server, it should be the only process on that port.

If you have one or more programs established or listening on the same port as the Cogent DataHub, you have two choices:

- Change the port number for the DataHub Web Server (as illustrated above), or
- Change the port number for every other program that is using that port.

Port numbers 1 through 1024 are reserved. Port 80, for example, is reserved for HTTP, which is why we make it the default for the DataHub Web Server. If you change the DataHub Web Server from port 80, we suggest setting it to a number between 1025 and 65535.

5. Configure any desired options, according to these guidelines:

**Error log file:**
The path and name of the file where errors are logged.

**Access log file:**
The path and name of the file where access attempts, successes, and failures are logged.

**SSL certificate file:**
The path and name of the certificate file used for secure sockets (SSL). Please see SSL Certificates for more information about SSL certificates in the DataHub.

**Require a password for all requests**
Applies password security as configured in Security.

The security model changed from version 7.1 to version 7.2. User names and passwords in the Security tab will be maintained when moving from V7.1 to V7.2. User names and passwords in the Web Server tab will be lost. When upgrading to V7.2 you must re-assign Web Server realms to any relevant users in the Security tab by clicking on the password field for that user. When reverting from V7.2 to V7.1, you must re-enter the path to the Web Server authentication file that you had earlier used with V7.1.
Using ASP to Query a Database and Display Results

Using the Cogent DataHub's ODBC scripting functionality, it is possible to send an SQL query from a web page to a database, and view the results in the page. The dbquery.asp demo page included in the DataHub distribution gives an example of how this is done. With the DataHub running on your local machine, you can view this page by typing localhost/dbquery.asp into your web browser:

You simply enter a DSN, user name, and password (if any) for the database, and an SQL query. When you press the Submit button, the page sends the query to the DataHub, which passes the query to the database, and returns the results, displaying them in the table on this page.

The source file for this page is included with all the other ASP files here in your DataHub distribution:

C:\Program Files (x86)\Cogent\Cogent DataHub\Plugin\Webserver\html\dbquery.asp
C:\Program Files\Cogent\Cogent DataHub\Plugin\Webserver\html\dbquery.asp

Here is a copy of the page source:

<html>
<head>
<%
/*
 * This file presents an entry form for the user to specify a
 * DSN, user name, password and SQL query to be executed by the
 * DataHub. The result is displayed as a table in the user's
 * browser.
 *
 * Normally, the DSN, user name and password would be hard-coded
 * into the Gamma portion of this ASP file. The ASP file is
 * stored on the server running the DataHub, and all Gamma code
 * is stripped from the file and executed before the result is
 * returned to the user. Therefore the DSN, user name and
 * password are never transmitted across the network to the user.
 */

require ("AJAXSupport");
require ("ODBCSupport");

local cmpfn, getArg, sortfn;

/* Set up some function for quickly accessing the URL arguments */
function cmpfn(x,y) { strcmp(x[0],y); }
function sortfn(x,y) { strcmp(x[0],y[0]); }
function getArg(name, dflt)
{
    local arg = bsearch(_vars, name, cmpfn);
    if (arg && !undefined_p(car(arg)))
        arg = car(arg)[1];
    else
        arg = dflt;
    arg;
}
_vars = sort(_vars,sortfn);

/* Read the input URL arguments */
local DSN = getArg("DSN","Enter DSN");
local Password = getArg("Password","Enter Password");
local Username = getArg("Username","Enter Username");
local Query = getArg("Query","");

/* Connect to the database */
local env, conn, Connect, DoQuery;

function Connect ()
{
using the web server

local ret;

/* Create the ODBC environment and connection */
env = ODBC_AllocEnvironment();
conn = env.AllocConnection();

/* Attempt the connection. */
ret = conn.Connect (DSN, Username, Password);
if (ret != SQL_SUCCESS && ret != SQL_SUCCESS_WITH_INFO)
    error (conn.GetDiagRec());
}

function DoQuery()
{
    local result = conn.QueryToTempClass(nil, Query);
}

<%>
</head>

<body>

Enter the database information:
<form>
DSN: <input type="text"
    id="DSN" name="DSN" value="<%= DSN %>">
User: <input type="text"
    id="Username" name="Username" value="<%= Username %>">
Pass: <input type="password"
    id="Password" name="Password" value="<%= Password %>">
</form>

Enter the database query to perform and press Submit:
<form>
<textarea name="Query" wrap="logical" rows="10" cols="80">
<%= Query %></textarea>
</form>

<% try {
if (Query != "")
    Connect();
}
local result = DoQuery();
if (!result || length(result) == 0)
{
}
else
{
    local first = result[0];
    %> <div style="overflow: auto; width: 660; height: 300"> <%
    %> <table border="1"> <%
    %> <tr> <%
        with var in instance-vars(first) do
        {
            %> <th><%= car(var) %></th> <%
        }
        %> </tr> <%
    with row in result do
    {
        %> <tr> <%
            with var in instance-vars(row) do
            {
                %> <td><%= cdr(var) %></td> <%
            }
            %> </tr> <%
        }
        %> </table> <%
        %> </div> <%
    }
} catch {
    %> A problem occurred while running server-side scripts on this page:<br>
        <%= _last_error_ %> <%
    }
/* Do a little cleanup. The garbage collector would eventually get to this, but we can be kind and unwind. */
if (conn)
{
    conn.Disconnect();
    destroy (conn);
}
if (env)
    destroy (env);
Generating and Receiving XML with the Cogent DataHub

The Cogent DataHub can exchange XML-formatted data with web-based clients in several ways. The choice of mechanism will be determined by the needs of the client application. Generally, these mechanisms can be categorized into either streamed or polled methods.

Streaming XML Data

The Cogent DataHub Web Server provides an efficient method for streaming data over a TCP/IP socket. The initial socket connection is negotiated using HTTP/HTTPS. Once the socket is established, a separate thread in the Cogent DataHub takes over responsibility for the socket. After the initial HTTP negotiation, all communication is uni-directional. That is, the DataHub emits data to the client application, but expects no data to be transmitted by the client. The XML format can be pre-defined or user-configured. The actual formatting of the data is performed via scripts running in the DataHub scripting language, Gamma.

The streamed XML mechanism makes use of the same underlying technology that provides the “Streaming AJAX” in the Cogent DataHub. Only the data format is different.

Polling XML Data

The Cogent DataHub Web Server provides two methods for receiving XML data via polling. In the common case, the Cogent DataHub offers a special URL that directly accesses the data set within the DataHub engine to construct an XML string. The XML format is pre-defined. The string construction is performed entirely in memory, so this method is very efficient.

If the client requires a different XML format, this can be provided via an ASP page. The DataHub uses Gamma as its ASP language, meaning that a request for a web page may trigger Gamma scripting calls that produce the XML data in any format that the web developer requires.

Streaming XML How-To

Built-in Streaming Data

The built-in streaming data is accessed using a URL of the form: http://host-name:port/stream?arguments. The arguments are separated by the & symbol, and can be any combination of:

name=pointname1|pointname2|...

This is a list of individual point names that will be retrieved from the DataHub. The
point names must be fully-qualified, such as `DataSim:Sine` rather than just `Sine`.
The point names are separated by the pipe character, `|`. If the name argument is
omitted then the domain argument must be provided. Example:

```
name=DataSim:Sine|DataSim:Ramp|DataSim:Square
```

template=[json,jsonp,djson,djsonp,xml,lisp]
This selects the output formatting from the available options shown in the list. The
template will determine the default head, tail, prefix and suffix. Example:

```
template=xml
```

default name
The head is the name of a file relative to the document root of the DataHub Web Ser-
er that will be transmitted once when the connection is first made. This can contain in-
formation like XML version or web page header information. Example:

```
head=my_header_file
```

default tail name
The tail is the name of a file relative to the document root of the DataHub Web Server
that will be transmitted immediately prior to terminating the connection. The connec-
tion will be terminated automatically by the DataHub when the msglimit is reached.
If msglimit is 0, this file will never be transmitted.

```
prefix=<any string>
```

The prefix XML tag is transmitted prior to every data update. The DataHub will trans-
m its data as it becomes available, subject to the throttle setting. If there is more than
one data value to be transmitted, the prefix is transmitted once for every group of da-
ta points, not once per data point. Example:

```
prefix=<points>
```

```
suffix=<any string>
```

The suffix XML tag is transmitted after every data update. See prefix above for more
details. Example:

```
suffix=</points>
```

msglimit=a non-negative integer
Some clients need to periodically close and re-open a streaming connection in order
to clear resources accumulated while the connection is open. This will allow the client
to indicate how many data updates can be transmitted before the DataHub must
close the connection. It is the client's responsibility to re-open the connection once it is
closed. If this value is 0, the Cogent DataHub will never intentionally close the connec-
tion. The DataHub will only transmit the tail file before closing the connection, so if
this value is 0, the tail file will never be transmitted. Example:

```
msglimit=10000
```

throttle=a non-negative floating-point number
The number of seconds to wait before sending data, allowing a client to request a
maximum update rate from the Cogent DataHub. That is, if data is changing faster
than throttle seconds then the DataHub will accumulate data points and transmit
them only after throttle seconds have passed since the previous transmission. If a data point changes more than once within this period, only the most recent value is transmitted. Set throttle to 0 to indicate that the Cogent DataHub should transmit all data without delay. Example:

```
throttle=0.25
```

**user=a non-empty string**

If the Cogent DataHub security settings require a user name and password for a TCP connection, the client can supply those here. Example:

```
user=my_name&pass=my_pass
```

**pass=a non-empty string**

See above.

**domain=a domain name list**

The client may request all of the data in a data domain instead of naming data points individually. If the domain argument is provided, the name argument should not be provided. Multiple domain names can be separated by the pipe character, "|". Example:

```
domain=DataPid
```

Due to a bug in the Cogent DataHub, this argument is not currently functional. Instead, use: name=domain_name&children=1&recursive=1. Example:

```
name=DataPid&children=1&recursive=1
```

**children=[0,1]**

When the client supplies a name argument, it can also supply the children argument to indicate whether to also retrieve any child points of that point. If multiple points are supplied in the name argument, then the children argument will apply to all points. A value of 0 indicates not to retrieve children. A value of 1 indicates to retrieve children. This is further affected by the recursive setting below. Example:

```
children=1
```

**recursive=[0,1]**

If the children argument is 1, then the value of recursive will determine whether to walk the data hierarchy starting at the named point, retrieving all descendants of that point. If the value of recursive is 0, then only the direct children of each point will be retrieved. If the value of recursive is 1, then all descendants of the named points will be retrieved. Example:

```
recursive=1
```

**Example**

To retrieve all of the points in the DataPid domain, at a maximum update rate of 5 Hz, for a maximum of 10000 data changes, the URL would be:

```
http://localhost/stream?name=DataPid&children=1&recursive=1
```
Using the Web Server

The resulting output would look like this:

```xml
<points>
    <point name="DataPid:PID1.Controller.Kd" value="0.01" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Controller.Ki" value="0.5" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Controller.Kp" value="0.25" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Mv" value="37.2139761632173" type="1" quality="192" timestamp="1275683572.9200001" />
    <point name="DataPid:PID1.Plant.Ki" value="0.5" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Plant.Kp" value="2" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Pv" value="55.551238388592" type="1" quality="192" timestamp="1275683572.9200001" />
    <point name="DataPid:PID1.Range.Amplitude" value="100" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Range.Offset" value="50" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Setpoint.AutoMode" value="1" type="2" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Setpoint.AutoTime" value="5" type="1" quality="192" timestamp="1275682823.9979999" />
    <point name="DataPid:PID1.Setpoint.SpInput" value="0" type="1" quality="192" timestamp="-2147483648" />
    <point name="DataPid:PID1.Sp" value="72.4715414899136" type="1" quality="192" timestamp="1275683572.1540003" />
    <point name="DataPid:PID1.UpdateFrequency" value="10" type="1" quality="192" timestamp="1275682823.9979999" />
</points>
<points>
    <point name="DataPid:PID1.Mv" value="38.3540008662675" type="1" quality="192" timestamp="1275683573.1379999" />
    <point name="DataPid:PID1.Pv" value="57.5623437038486" type="1" quality="192" timestamp="1275683573.1379999" />
</points>
<points>
    <point name="DataPid:PID1.Mv" value="39.2907100148899136" type="1" quality="192" timestamp="1275683572.1540003" />
    <point name="DataPid:PID1.Pv" value="59.5695627360927" type="1" quality="192" timestamp="1275683573.3569999" />
</points>
```
Customizing the Built-in Streaming Data

The format of streaming web data is controlled through a Gamma script provided in require\WebstreamSupport.g in the DataHub installation directory. You can add your own format templates to allow you to specify how the data is presented to your program. For example, the default XML layout looks like this:

```xml
<points>
  <point name="DataSim:Sine" value="0.5" type="1" quality="192" timestamp="1275683573.7950006" />
</points>
```

There is no head or tail string. The prefix string is `<points>` and the suffix string is `</points>`. In addition there are three available format strings to determine how to format different data types:

- **stringformat** - a format string that will be used to prepare string data
- **numberformat** - a format string that will be used to prepare numeric (real and integer) data
- **arrayformat** - a format string that will be used to prepare array data

In the default XML template, these are:

```xml
stringformat - <point name="%s" value="%w" type="%d" quality="%d" timestamp="%g" />
numberformat - <point name="%s" value="%s" type="%d" quality="%d" timestamp="%g" />
arrayformat - <point name="%s" value="%s" type="%d" quality="%d" timestamp="%g" />
separator - a string used to separate multiple points between a single set of prefix and suffix strings. E.g., ",,"
```

Each of the format strings must encode 5 input parameters, supplied in order as:
name - a string representing the full point name
value - the value of the point. This will be one of string, number or array
type - a number representing the data type, where 0 = string or array, 1 = floating point number, 2 = integer number
quality - an OPC quality as an integer number
timestamp - a UNIX time stamp as the number of seconds since January 1, 1970 UTC

Arrays are encoded as strings of the form "['value1','value2','value3',...]".

These parameters are specified by modifying the definition of the Gamma class WebstreamSupport. The original definition is found in the file require\WebstreamSupport.g. It is wise not to modify this file, but instead to add your modifications in a separate file as follows:

1. Create your own Gamma file, e.g., MyCustomWebstream.g.
2. Remove all existing code in this file
3. Insert the following code:

```csharp
require ("WebstreamSupport");

// Ensure that there is an instance variable for my new template
if (!has_ivar(WebstreamSupport, #my_custom))
{
    class_add_ivar(WebstreamSupport, #my_custom);

    // Re-instantiate the helper instance with the new
    // instance variable included
    Webstream = ApplicationSingleton (WebstreamSupport);
}

Webstream.my_custom = new WebstreamTemplate ("CUSTOM", "", "my_custom",
"<data>
"</data>\n",
0, 0.0,
"<?xml version="1.0" encoding="UTF-8"?>\n",
",
"<point name="%s" value="%w" type="%d" quality="%d"
"timestamp="%g" />
",
"<point name="%s" value="%s" type="%d" quality="%d"
"timestamp="%g" />
",
//"<point name="%s" value="%s" type="%d" quality="%d"
"timestamp="%g" />
",
",
)
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
4. Modify the code to match your requirements.

The code creates a new template member in the class `WebstreamSupport` and then assigns it a set of parameters by creating a `WebstreamTemplate`. The constructor to `WebstreamTemplate` takes the following arguments:

- `tplname` - an arbitrary string. It is not used.
- `pointnames` - a string of pipe-separated default point names, should be ""
- `template_id` - the name of the template supplied in the /stream URL
- `prefix` - a string that is prefixed to every group of data changes
- `suffix` - a string that is suffixed to every group of data changes
- `msglimit` - default msglimit as above
- `throttle` - default throttle as above
- `head` - a head string (not a file name as above) that is transmitted once at connection
- `tail` - a tail string that is transmitted immediately prior to the DataHub disconnecting
- `stringformat` - as above
- `numberformat` - as above
- `arrayformat` - as above. Depending on your version of Cogent DataHub, this argument may be missing.
- `separator` - as above

5. Ensure that your new script runs when the Cogent DataHub starts.

6. Run your script. You should now be able to query data from a URL like this:

```
http://localhost/stream?name=DataPid&children=1&recursive=1
&template=my_custom
```

### Polling XML How-To

Typical Ajax applications retrieve data through polling, using a Javascript function called `XmlHttpRequest`. This function makes an asynchronous HTTP request to a URL that returns an XML document. The Javascript application parses this XML document to extract relevant information.

### Built-in HTTP Data Polling

The Cogent DataHub implements a built-in XML polling mechanism that constructs the XML document entirely in memory to minimize CPU usage. The format of this document is not user-configurable, and is similar to the output produced by the streaming data facility when specifying `template=xml`.
The built-in streaming data is accessed using a URL of the form: http://host-name:port/points?arguments. The arguments are separated by the & symbol, and can be any combination of:

```
name=pointname1|pointname2|...
```

This is a list of individual point names that will be retrieved from the DataHub. The point names must be fully-qualified, such as DataSim:Sine rather than just Sine. The point names are separated by the pipe character, "|". If the name argument is omitted then the domain argument must be provided. Example:

```
name=DataSim:Sine|DataSim:Ramp|DataSim:Square
```

```
domain=a domain name list
```

The client may request all of the data in a data domain instead of naming data points individually. If the domain argument is provided, the name argument should not be provided. Multiple domain names can be separated by the pipe character, "|". Example:

```
domain=DataPid
```

Due to a bug in the Cogent DataHub, this argument is not currently functional. Instead, use: name=domain_names&children=1&recursive=1. Example:

```
name=DataPid&children=1&recursive=1
```

```
style=[json,jsonp,djson,djsonp,xml,lisp]
```

This selects the output formatting from the available options shown in the list. Example:

```
style=xml
```

```
prefix=<any string>
```

The prefix XML tag is transmitted prior to every data update. The DataHub will transmit data as it becomes available, subject to the throttle setting. If there is more than one data value to be transmitted, the prefix is transmitted once for every group of data points, not once per data point. Example:

```
prefix=<points>
```

```
suffix=</any string>
```

The suffix XML tag is transmitted after every data update. See prefix above for more details. Example:

```
suffix=</points>
```

**Customized HTTP Data Polling**

Since the built-in data polling is not user-configurable, it is not an appropriate mechanism for generating custom data formats. Instead, customized XML data can be emitted using a user-supplied ASP page. The ASP mechanism within the Cogent DataHub Web Server uses the Gamma scripting language as its ASP language. This means that an ASP page has access to the live data in the DataHub and any Gamma functions such as database calls. The Gamma calls can be used to generate strings that are inserted into the page.
A simple ASP page that generates an XML may look like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<points>
  <point name="DataSim:Sine" value="<%= $DataSim:Sine %>">
</points>
```

In this example, the structure of the document is simple XML, with data from Gamma calls embedded within `<%= %>` delimiters. The result of evaluating the Gamma expression is converted to a string and then inserted into the file, replacing the `<%= %>` delimiters and all text between those delimiters. The Gamma script between these delimiters must be a single Gamma expression, not a statement. For example, the expression `2+2` is legal, but the statement `a=2+2;` is not.

A more complex example could implement a loop in Gamma to query the data set in the DataHub and provide a value for every data point.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<points>
  <local points = datahub_points("DataPid", nil);
  with point in points do
  {
    if ((point.flags & 0x30) == 0) // ensure point is not an assembly
    {
      <point name="<%= point.name %>">
        value="<%= point.value %>">
    }
  }
</points>
```

In this example, the delimiters `<%= %>` are used to indicate Gamma script code that will not generate a replacement string. The Gamma script between `<%` and `%>` must be one or more Gamma statements, not expressions. Notice that Gamma statements may be broken up across occurrences of the `<% %>` delimiters, allowing a natural mechanism for specifying the XML portion of the file.

The ASP file can be placed in any subdirectory of the document root of the DataHub Web Server. The client application can then simply make repeated reads of this file to retrieve the current data values.

**ASP File Locations**

As a security precaution, ASP files can only be loaded from specific base URL locations. You can add more URL paths as necessary. By default, ASP pages can only be loaded from the following URL locations:
These are URL paths, not file system paths. URL paths are determined relative to the document root path. In the DataHub, there are in fact two document roots, the Web Server root and the user content root. The Web Server root contains files that are typically common to all DataHub instances, and the user content root contains user-generated content for a particular DataHub instance. If an identically named document exists in both roots, the document in the user content root will take precedence.

Web Server Document Root

The Web Server document root is defined in the DataHub Web Server properties dialog. This is normally the `html` folder within the DataHub installation folder. The files in this folder are not intended to be modified by the user. If you re-install the DataHub, it will overwrite files in this folder.

The default Web Server root for 32-bit DataHub is:

```
C:\Program Files (x86)\Cogent\Cogent DataHub\plugin\WebServer\html\n```

The default Web Server root for 64-bit DataHub is:

```
C:\Program Files\Cogent\Cogent DataHub\plugin\WebServer\html\n```

This default path can be reconfigured in the Properties window Web Server option, if desired.

The ASP file locations are folders named after one of the URL paths, appended to the document root. For example, in the 64-bit DataHub, the installed ASP files are located relative to the Web Server document root:

```
C:\Program Files\Cogent\Cogent DataHub\plugin\WebServer\html\Silverlight
C:\Program Files\Cogent\Cogent DataHub\plugin\WebServer\html\PointList
```

User Content Document Root

There is a second document root for user-created content. This root is the WebContent folder, located in the DataHub configuration folder, here by default:

```
C:\Users\UserName\AppData\Roaming\Cogent DataHub\WebContent\n```

User-created ASP files must go into one of the folders named after a URL path, appended to this root:

```
C:\Users\UserName\AppData\Roaming\Cogent DataHub\WebContent\Silverlight
C:\Users\UserName\AppData\Roaming\Cogent DataHub\WebContent\PointList
```
You can change the default DataHub configuration folder using the \(-H\) command-line option, or by setting it through the DataHub Service Manager application. If you make that change, you must also move all user-created ASP files to correctly-named folders in the new configuration folder.

**Adding ASP folders**

You can add ASP folders by adding URL paths to the three in the list above. This is done by hand-editing the `plugin_AspHandler.cfg` file in your DataHub configuration folder. Just copy any `UrlRule` line and modify the `BasePath` to the path that you want to serve ASP pages from.

If you do not find the `plugin_AspHandler.cfg` file, you can create one, using this content as the starting point:

```xml
<?xml version="1.0" encoding="utf-8"?>
<AspHandlerSettings xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <UrlRule BasePath="/" IncludeSubdirs="false"
        ByOrganization="false" RuleResult="Allow"/>
    <UrlRule BasePath="/Silverlight" IncludeSubdirs="false"
        ByOrganization="false" RuleResult="Allow"/>
    <UrlRule BasePath="/PointList" IncludeSubdirs="false"
        ByOrganization="false" RuleResult="Allow"/>
</AspHandlerSettings>
```

When you add a new `UrlRule` to this file, it will allow ASP pages to be loaded from an identically named folder in either of the two roots: the Web Server document root, or the user content document root.
Using MQTT

MQTT is a publish/subscribe messaging protocol. It allows data sources and users called clients to connect to a server called a broker, and exchange their data through it. Clients can act as both publishers and subscribers simultaneously. The DataHub provides both MQTT client and MQTT broker functionality.

Because it is a messaging protocol, MQTT is different from a data communications protocol. MQTT acts as a data transport layer, similar to TCP, but it does not specify a particular format for the data payload. The message format is specified by a client when it connects. Any clients that want to connect to each other must use the same message format. The broker, of course, must also use that same message format.

Connecting MQTT Clients with the DataHub MQTT Broker

The DataHub MQTT Broker offers two modes of operation: standard and gateway.

Standard MQTT Broker Mode

In standard MQTT broker mode the DataHub MQTT Broker simply passes messages from one client to another, and the messages remain completely independent of the DataHub's data set.

To configure the DataHub MQTT Broker as a stand-alone broker, simply select the Do not process messages, just route them option in the Message Content section of the MQTT Broker configuration.

Gateway Broker Mode

In gateway mode you can integrate MQTT messages with the DataHub data set, as normal DataHub data points. Here is how you configure the DataHub for each of these modes. There are three options for gateway mode:

• Binary messages are Base-64 encoded and stored in DataHub points as strings.
• Text messages are interpreted as UTF-8 text stored in DataHub points as strings.
• JSON messages get interpreted by the DataHub as data point values that can have associated quality, timestamps, and so on.

For more information about these options, please refer to Message Content for the MQTT Broker.

Changing the JSON client message format

The DataHub's MQTT Broker's default JSON message format looks like this:
This JSON format allows the DataHub to assign an MQTT topic to a data point, and send or receive a value, quality, timestamp, and sender ID for that point. To connect a client that uses a different JSON message format, you need to change this default format for that client’s format, as follows:

1. With the DataHub running, right click on the DataHub system-tray icon and choose Properties.
2. In the Properties window, select MQTT Broker.
3. In the Message Content section ...

press the Edit button to open the Configure Parser window:
If you know the client message format

1. In the **Per-Point Format** field, enter your MQTT client message format.
2. Click **OK** and then **Apply** to save your settings.
3. Make any other desired changes to the MQTT Broker configuration as described in the section called “MQTT Broker”.

The DataHub MQTT Broker is now configured for your MQTT client.

If you want to accept any client message

1. In the **Per-Point Format** field enter just the string: `{value}`.
2. Delete any entries in the **Message Start**, **Per-Point Separator**, and **Message End** fields.
3. Click **OK** and then **Apply** to save your settings. Your **Message format** should now look like this:
Now the DataHub will accept any client message, parsing it as follows:

- Numbers without decimal points become integer values.
- Numbers with decimal points become real values.
- Everything else becomes a text string.

**If you want to discover and use the client message format**

If you don't know the client message format, but want to use it as the DataHub's MQTT message format, you can use the `{value}` message format to discover it, as follows:

1. Ensure that the DataHub’s MQTT Broker is configured with the `{value}` message format, as described above.
2. Start your MQTT client, connect to the MQTT Broker, and send a message. For example, here is a message that is not the DataHub’s default message format, but can write to the point MQTTBroker:DataPid.PID1.Mv in DataHub.

   ```json
   [{ "VAL": 33, "TPC": "DataPid/PID1/Mv", "TS": 1548276263.866, "QTY": 192, "ID": "" }]
   ```

3. When the DataHub's MQTT Broker receives this message, you will be able to see the entire, unparsed string as the value for a point in the Data Browser. The point is in the domain that you have configured for the MQTT Broker.

4. Click on the point name to display the value in the **Enter new value** field.
5. Copy the complete string and paste it into the **Per-Point Format** field of the Configure Parser window of the DataHub MQTT Broker configuration.
6. Remove the outer pair of square brackets ([ and ]) from the string, and replace the values of the parameters with the placeholder values, like `{topic}`, `{value}`, `{quality}`, etc.
7. Restore the entries for Message Start ( ), Per-Point Separator ( , ), and Message End ( ) fields.
8. Click OK and then Apply to save your settings.
9. Make any other desired changes to the MQTT Broker configuration as described in the section called “MQTT Broker”.

The DataHub MQTT Broker is now configured for your MQTT client. If you push a message from the client with a new value, that value should appear in the Data Browser.

To make the MQTT connection, use the IP address of the computer running the DataHub, and the port that you have configured. The defaults are 1883 for plain text and 8883 for SSL.

The DataHub MQTT Broker does not support WebSocket connections for MQTT.

**Making MQTT Client Connections**

The DataHub MQTT Client option lets you connect with virtually any MQTT broker, to publish or subscribe to any or all topics. If you are using Azure IoT Hub, Google IoT Hub, or Amazon IoT Core, please see those sections for configuring client connections. To connect to any other MQTT broker, here is what you need to do.

To keep things simple, if you plan to run an MQTT broker on your local machine, ensure that the DataHub MQTT Broker is not also running. You can disable it by unchecking the Enable MQTT Broker box in the MQTT Broker configuration.
Subscribe (get topics)

Before pushing any data to the MQTT broker, it can be useful to know some or all of its topics. This procedure shows how to register the DataHub with an MQTT broker and bring its topics into the DataHub MQTT client and the DataHub data set.

1. In the DataHub Properties window, select **MQTT Client**.

2. Click the **Add** button to open the Connect to MQTT Broker window.

3. In the **Standard MQTT** tab, for the **Connection** enter the following:

   - **Label**: Enter any text string.
   - **Host**: Enter the name or IP address of the host of the broker.
   - **Port**: 1883 (the default)
   - **Keepalive**: 30 (the default)
   - **Retry rate**: 5000 (the default)
   - **Maximum update rate**: 0 (the default).

4. In the **Message Content** tab ...
Using MQTT

press the **Edit Format** button to open the Configure Parser window:

5. In the **Per-Point Format** field enter just the string: `{value}`.

6. Delete any entries in the **Message Start**, **Per-Point Separator**, and **Message End** fields.

7. Click **OK**. Your **Message format** should now look like this:
8. Choose **Pull topics from the MQTT Broker**.

![MQTT Broker option]

9. Click the **Add** button and enter the hash symbol (#), which tells the broker that you want to subscribe to all topics.

10. Check the **Place all data points into this data domain** box, and enter **MQTTClient** in the entry field.

![Data domain settings]

This keeps all of your MQTTClient points separate from other DataHub data.

11. Click **OK** and then **Apply** to save your settings.

12. Open the DataHub Data Browser. You should now see all of the topics from the MQTT Broker in the **MQTTClient** domain.

![Data browser]

If the broker's payload format does not map directly to DataHub points, you may wish to change it. Please see **Message Payload Formats** for more details.

**Subscribe (push data)**

This procedure shows how to push data from DataPid to your MQTT broker.

1. In the Properties window, select **MQTT Client**.

![MQTT Client option]
2. Click the **Add** button to open the Connect to MQTT Broker window:

3. In the **Standard MQTT** section, **Connection** tab, enter the following:
   - **Label**: Enter any text string.
   - **Host**: Enter the name or IP address of the host of the broker.
   - **Port**: 1883 (the default)
   - **Keepalive**: 30 (the default)
   - **Retry rate**: 5000 (the default)
   - **Maximum update rate**: 0 (the default).

4. In **Authentication**, enter the **Username** and **Password** for the MQTT broker, if applicable. There is no need to change the default settings in **Message Content** or **Last Will Message**.

   This example uses the DataHub's default JSON message content format. If you want to use a different format to send data to a broker, please refer to the client **Message Content** documentation for how to change it. If you want to simply send the unparsed values, please refer to the **Unparsed Values** documentation.

5. In the **Push data points to the MQTT broker** section, **Available Points** list, open the **DataPid** tree and select the point **Pv**. (If you don't see the **DataPid** domain, start **DataPid**.)
6. Click OK, and then **Apply**.

7. In the MQTT broker or a connected MQTT client, you should see the values for **DataPid:PID1.Pv** updating. These messages are using the DataHub's default message content format.

You have now configured the DataHub MQTT Client to push a value to an MQTT Broker. For more information about the MQTT Client feature, please refer to the section called “MQTT Client”.

**Subscribe (pull data)**

With data from DataPid being pushed to your MQTT broker, you can now create an MQTT client connection to pull that topic and its data from the broker.

If you have not yet completed the previous procedure, please do it now for best results here.

1. In the Properties window, select **MQTT Client**.

2. Click the **Add** button to open the Connect to MQTT Broker window:
3. In the **Standard MQTT** section, **Connection** tab, enter the following:
   - **Label:** Enter any text string, different from any previous label.
   - **Host:** Enter the name or IP address of the host of the broker.
   - **Port:** 1883 (the default)
   - **Keepalive:** 30 (the default)
   - **Retry rate:** 5000 (the default)
   - **Maximum update rate:** 0 (the default)

   There is no need to change any settings in **Authentication**, **Message Content**, or **Last Will Message**.

   These settings assume that the broker is using the DataHub's default JSON message content format. If your broker is using a different format, please refer to the **Message Content** documentation for how to change it. If you want to simply receive unparsed values from the broker, please refer to the **Unparsed Values** documentation.

4. In the **Pull topics from the MQTT broker** section, click the **Add** button and enter DataPid/PID1/Pv.

5. Click the **Place all data points into this data domain** button and enter a domain name, such as **MQTTPullTest**.
6. Click OK, and then Apply.

7. In the Data Browser, you should see the values for DataPid:PID1.Pv updating, as normal DataHub point values.

You have now configured the DataHub MQTT Client for a topic to pull values from an MQTT broker. For more information about the MQTT Client feature, please refer to the section called “MQTT Client”.

Message Payload Formats

Messages for or from an MQTT broker might not have a payload format that maps directly to DataHub points. Data points in the DataHub are 4-element tuples of the form (name, value, quality, timestamp). Those are the only elements that can be extracted from the information being sent to or arriving in an MQTT payload for the DataHub. In the JSON format, these rules will be applied:

- Only one of topic or point will be used.
- Only one timestamp will be used.
- Only one of quality or qualityname will be used.

A payload that you want to send or receive may contain more information. If you need any information other than name, value, quality, and timestamp, you need to handle it yourself. To send this information or bring it into the DataHub, there are 2 choices:

1. Create a point value as a text string that contains all of these in a format you define, like JSON or CSV.

2. Create different data points for each of these items, like mypoint.data1, mypoint.data2, etc.

In either case, you will need to set the per-point format to {value}. That will preserve the whole MQTT payload, and that's all that's needed for the first case. For the second case, you will need to write a script to construct the payload or break it apart, and write the re-
results as separate individual points in the DataHub. If you do both, then any DataHub client will have the choice of receiving/collecting the entire payload in the original point, or sending/looking at the different points that the script creates.
Security

The Cogent DataHub provides a means for full access control to all DDE, TCP, OPC, and tunnel/mirror connections, using authentication and authorization. Authentication limits access to recognized users, based on a username/password combination. Authorization provides a set of permissions for each user, allowing access to certain functions while denying access to others.

The DataHub also provides full SSL (Secure Sockets Layer) encryption for TCP/IP tunnelling and mirroring connections.

How to Configure

In the DataHub Properties window, select Security.

Click the Configure Permissions button to open the Edit Permissions window.

Here you can create and modify groups, and then assign users to those groups.
Groups

Groups provide a convenient way to configure a number of users who have identical permissions. Each group can be assigned a unique set of permissions from the Permissions table. There are several default groups, such as BasicConnectivity, HTTPUser, Admin, and RemoteConfig. To add a group, type a group name in the bottom row of the Groups table. Check or uncheck the boxes to assign permissions.

For example, in the illustration below an Operator has been added that has been given Connection permissions for Connect, Read, and Write.

Users

There are two kinds of users: normal and special. Normal users correspond to individuals with a name and a password. Special users provide a way to offer different security models for different protocols. For more information on types of users, please refer to the section called “User Authentication”.

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
To add a user, type a user name in the bottom row of the **User** table. When you press **Enter**, a password dialog will appear:

Enter a password and select an HTTP realm for that user. When you click **OK**, a string of characters will appear in the **Password** field for that user. Passwords are stored using a reasonably strong non-reversible encryption. If a user forgets his password, it is not recoverable. To change HTTP realm for a user, their password must be reentered. For more information on passwords, please refer to the section called “Passwords”.

**HTTP Realms**

Here you can maintain a list of HTTP authentication realms. This list is accessed by the DataHub Web Server, as described in the section called “Web Server”.

---

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
To add an authentication realm, simply type it into the list. One or more of these realms are assigned to each user when their password is configured (see above).

**Common Scenario**

The most common Cogent DataHub security configuration is to allow any user to connect via OPC or DDE, while only allowing authorized users to connect via TCP or via a tunnel/mirror. This eliminates exposure of the TCP and tunnel/mirror connections to unwanted Internet and network clients. OPC and DDE are not exposed in this way.

To configure this scenario, you need to remove all group memberships from the special Anonymous, TCP, and Mirror users. Simply click on each of these user names in turn, and uncheck all group memberships for that user. When you are finished, only DDE and OPC should have any group memberships.

**SSL and Firewalls**

The Cogent DataHub provides the option to use SSL encryption to protect your data when configuring Tunnel/Mirror to another DataHub across a network connection, or for anyone using WebView or the Web Server. The SSL implementation uses the default SSL-3 encryption cipher: DHE-RSA-AES256-SHA, which is a 256-bit encryption method.

**SSL Certificates**

The DataHub offers SSL encryption both for Tunnelling and for making HTTPS connections to the DataHub Web Server. An SSL certificate is required to use SSL encryption on the DataHub. A default SSL certificate is installed with the DataHub, or you can use your own certificate.

The certificate in the DataHub installation is a self-signed certificate with an expired end-date. This makes the certificate invalid when checked by the client side of the connection, which is intentional. It is impossible for us to ship a valid certificate. A valid certificate for HTTPS must be issued for the particular domain name or IP address that is hosting the server, and we cannot predict what that will be on your system. In addition, if we ship a certificate it will contain the private key, and anybody else who downloads and installs the DataHub will also have the same private key. This means that using a certificate that is bundled into the DataHub installer will not protect you from malicious attacks—it will not provide reliable identification of the server, and it will not provide trustworthy encryption. Anybody with access to the certificate and Wireshark can decrypt the entire SSL conversation.

The reason we provide the DataHub self-signed SSL certificate is so that you can easily make encrypted Tunnelling and HTTPS connections from within a trusted environment,
where you require encryption but don't want to manage and renew certificates on each of your DataHub installations. Because the SSL certificates are checked by the client side of the connection (the side that is initiating the connection), we provide options in the DataHub to ignore invalid certificates, which in turn allows you to use the self-signed SSL certificate we ship with the DataHub. For Tunnelling, these options are in the configuration of the Tunnel Slave connection. For applications that connect to the DataHub Web Server such as WebView and Remote Config, the options to ignore invalid certificates are provided on the login screens where you enter your username and password.

If you are working in an untrusted environment and are required to have a valid SSL certificate on the DataHub, then you must either generate your own certificate using a trusted certificate authority, or get a certificate from a trusted public certificate authority like Verisign or Comodo. In this case you would then configure the DataHub to use your trusted certificate and you would disable the options to ignore invalid certificates when making a WebView, Tunnel or Remote Config connection.

**Firewall Ports**

The DataHub lets you specify which ports it will use for tunnelling/mirroring over a network. Firewalled ports can be secured, because if you open a port on the firewall, any program that attempts to connect on this port will need to be able to communicate with the DataHub that is listening on that port. As long as authentication is used for tunnelling, even a user who attempts to connect using another DataHub program will need to have access to a valid username and password.

**Modifying SSL Security Levels**

The SSL libraries for the DataHub are upgraded whenever new ones become available. If you are connecting different versions of the DataHub, at some point you may discover that a DataHub using an older version of SSL is no longer compatible, and cannot connect to a DataHub that uses a newer version of SSL. Should this happen, we encourage you to upgrade all instances of the DataHub to the latest release.

However, we realize that upgrading existing installations is not always an option. In these cases it may be possible to configure a tunnel connection by modifying the SSL security level of the newer version of the DataHub. This is done by changing a registry setting on the machine running that DataHub, as follows:

1. Open the registry key, located as follows:
   - For 64-bit Windows:
     ```plaintext
     HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Cogent\Cogent DataHub
     ```
   - For 32-bit Windows
     ```plaintext
     HKEY_LOCAL_MACHINE\SOFTWARE\Cogent\Cogent DataHub
     ```
2. Add a new **STRING** value named **SslMethod**.
3. Modify this value to one of these strings:
   - Tls1.2
• Tls1.1
• Tls
• SslV3
• SslV23

4. Restart the DataHub and re-try the connection.

You should try the suggested values in descending order until one works. Both SslV3 and SslV23 are considered very insecure. As of this writing, Tls, Tls1.1 and Tls1.2 are all considered secure.

If you select either SSL3 or SSL23 you should see a message in the DataHub event log similar to this:

Insecure SSL method SslV3 selected in registry.

Many remotely exploitable vulnerabilities have been identified in the older OpenSSL libraries, so if SSL security is important for this connection, you should seriously consider upgrading to the latest version of the DataHub.

User Authentication

Authentication of users is applied on a per-connection basis. This means that whenever a client program connects to the Cogent DataHub, it must transmit a user name and password in order to authenticate. Until the client program authenticates, it operates with the permissions of the anonymous user (see below). After 5 seconds, the permissions currently in force for the client are checked for the Connect permission. If the client does not have Connect permission, the connection is terminated. The client may authenticate as another user at any time after it has connected. If a client transmits an incorrect user name or password, it is immediately disconnected.

In early versions of the DataHub, the client was not immediately disconnected if an incorrect user name or password were entered. Instead, the client kept the permissions that were in force prior to the authentication attempt. As of version 8, this is no longer true.

Special Users

To facilitate special connectivity needs, the Cogent DataHub has several special users. The anonymous user represents a client that has not authenticated. When a client first connects, it is given the permissions of the anonymous user. The client may continue to operate with the anonymous user permissions (so long as the anonymous user has the Connections Connect permission), or may authenticate as another user at any time. In essence, the security of the DataHub is no greater than the permissions given to the anonymous user. The default distribution of the DataHub has anonymous user permissions enabled.
In addition to the anonymous user, there are special users associated with each connection protocol. These are essentially anonymous users that are associated with just one particular protocol. The protocols are:

- **DDE**: Any connection made from a DDE client to the Cogent DataHub.
- **OPC**: Any connection made from an OPC client to the Cogent DataHub.
- **TCP**: Any connection made from a third-party program using a direct TCP connection, the DataHub API, or a Java applet embedded in a web browser.
- **Mirror**: A mirror or tunnel connection from another Cogent DataHub

When a client connects using one of the above protocols, it is originally given the anonymous user permissions, and then promoted to the protocol user associated with the connection type, once the connection is fully constructed. This allows the Cogent DataHub to apply different permissions to anonymous connections of different types.

**OPC users**: Since the OPC Classic protocols (OPC DA and A&E) do not provide a mechanism for authentication, this is the only mechanism available to limit the permissions of an OPC Classic client.

### Normal Users and Groups

Clients to the Cogent DataHub are referred to as *users*. A user name is any combination of letters, numbers and some punctuation characters. A *password* can be any sequence of characters. Each user has an associated set of *permissions*. When a client transmits a correct user name and password, it acquires the permissions of that user.

Users can be assigned to a *group* to simplify the configuration of many users who have identical permissions. A user can be added to a group at any time. When added to a
group, the user's permissions will be altered to match those of the group. If the permissions for the group are subsequently changed, the change will immediately affect all users in the group. A user may belong to multiple groups.

Authorization and User Permissions

Every client program connected to the Cogent DataHub is associated with exactly one user at any given time. Each user is authorized to access certain features of the DataHub according to its user permissions. When a client first connects, it is immediately associated with the anonymous user, and gets those permissions. Then it gets switched to the special user for the protocol it is connecting on. If the client subsequently authenticates itself as a normal user, it is then granted that user's permissions. A client's permissions are always the entire permission set for the user that it is currently associated with.

To edit user permissions select the user name in the Users list and press the Add button. This will open the Permission Editor.

Permissions are categorized into four groups, and defined as follows:

Admin

ConfigAdmin
This user can remotely administer the application.

DeletePoint
This user can delete an existing point.

DeletePoint
Normally, no client should be allowed to delete points from the Cogent DataHub. Deleting points can be very disruptive for existing clients. Use this permission with caution.

LoadConfig
This user can load a configuration file.

Shutdown
This user can remotely terminate the application.

UserAdmin
This user can remotely administer other user configurations.

Connection

Connect
This user is allowed to maintain a connection to the Cogent DataHub. When a connection is made, the client has a 5-second grace period in which to authenticate before the client is disconnected. If the client does not have Connection Connect permissions after the grace period expires, it will be disconnected.

Data

ChangeModel
This user can change the organizational hierarchy of the data points. This is the infor-
uation that makes the data appear as a tree structure in the Data Browser.

**CreateDomain**
This user is allowed to create new data domains. Normally you should also set **Connection CreatePoint** permission when you set this permission for a user.

**CreatePoint**
This user is allowed to create new points in existing data domains in the Cogent DataHub.

**Force**
If the user has **Connection Write** permission, he may also have this permission. In this case, the user will able to send the **force** and **cforce** commands to the DataHub, which will override the read-only status and timestamp check for a point, thereby forcing a write to succeed where it would otherwise fail.

**HistoryRead**
This user is allowed to read historical values.

**Read**
This user is allowed to read point values and subscribe to point value changes.

**Write**
This user is allowed to write a new value to a DataHub point.

**HTTP**

**Connect**
This user is allowed to make an authenticated HTTP connection.

**Restrictions**

Beginning with DataHub v9.0, these six restriction permissions are not available. They are documented here only for backward compatibility.

**ConcurrentLoginLimit**
This restriction was never implemented.

**Expires**
This restriction was never implemented.

**ExpiryDate**
The date on which the user account expires, in YYYY-MM-DD format.

**LimitConcurrentLogins**
This restriction was never implemented.

**LimitTotalLogins**
This restriction was never implemented.

**MaxLoginLimit**
An integer specifying the maximum total number of logins permitted.
Web

CreateControl
   This user can create a new control (not implemented).

CreateMobilePage
   This user can create a new page for a mobile client.

CreatePage
   This user can create a new page.

DeleteControl
   This user can delete a control that he has created (not implemented).

DeleteOtherOwnerControl
   This user can delete a control that another user has created (not implemented).

DeleteOtherOwnerPage
   This user can delete a page that another user has created (not implemented).

DeletePage
   This user can delete a page that he has created (not implemented).

EditOtherOwnerControl
   This user can edit a control that another user has created (not implemented).

EditOtherOwnerPage
   This user can edit a page that another user has created.

EditPage
   This user can edit a page that he has created.

SaveControl
   This user can save a control that he has created (not implemented).

SaveOtherOwnerControl
   This user can save a control that another user has created (not implemented).

SaveOtherOwnerPage
   This user can save a page that another user has created.

SavePage
   This user can save a page that he has created.

ViewOtherOwnerPage
   This user can view a page that another user has created.

ViewPage
   This user can view a page that he has created.

WebView

BrowseInternet
   This user is allowed to browse the Internet from within DataHub WebView.

ChangeOptions
   This user is allowed to change DataHub WebView options.
**ChangeTheme**
This user is allowed to change the DataHub WebView theme. This function is not currently available.

**ConfigureTraceSettings**
This user is allowed to configure trace settings in DataHub WebView.

**Connect**
This user is allowed to make a live data connection from a DataHub WebView client.

**Troubleshoot**
This user is allowed to use the troubleshooting abilities of DataHub WebView.

**ViewOnlineHelp**
This user is allowed to view the DataHub WebView online help.

### Permissions for the DataHub Command Set

Each time the DataHub receives a command from a client, it checks the client's user permissions. Before executing the command, the DataHub compares the user's permissions to the permissions required to run the command (shown in the table below). If the user has the necessary permissions, the command is executed, otherwise an error message is returned.

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Permissions Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>acksuccess</td>
<td>none</td>
</tr>
<tr>
<td>add</td>
<td>Data / Write</td>
</tr>
<tr>
<td>alive</td>
<td>none</td>
</tr>
<tr>
<td>append</td>
<td>Data / Write</td>
</tr>
<tr>
<td>assembly</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>attribute</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>auth</td>
<td>none</td>
</tr>
<tr>
<td>authgroup</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>authuser</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>auto_create_domains</td>
<td>Change auto domain creation</td>
</tr>
<tr>
<td>auto_timestamp</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>bandwidth_reduce</td>
<td>none</td>
</tr>
<tr>
<td>bridge</td>
<td>Admin / ConfigAdmin, Data / Write</td>
</tr>
<tr>
<td>bridge_remove</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>bridge_transform</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>cforce</td>
<td>Data / Write, Force</td>
</tr>
<tr>
<td>cread</td>
<td>Data / Read, CreatePoint</td>
</tr>
<tr>
<td>create</td>
<td>Data / CreatePoint</td>
</tr>
<tr>
<td>Command Name</td>
<td>Permissions Required</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>create_domain</td>
<td>Data / CreateDomain</td>
</tr>
<tr>
<td>report</td>
<td>Data / Read, CreatePoint</td>
</tr>
<tr>
<td>cset</td>
<td>Data / Write, CreatePoint</td>
</tr>
<tr>
<td>cwrite</td>
<td>Data / Write, CreatePoint</td>
</tr>
<tr>
<td>debug</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>defaultprop</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>delete</td>
<td>Connection DeletePoint</td>
</tr>
<tr>
<td>deleted</td>
<td>Connection DeletePoint</td>
</tr>
<tr>
<td>div</td>
<td>Data / Write</td>
</tr>
<tr>
<td>domain</td>
<td>none</td>
</tr>
<tr>
<td>drop_license</td>
<td>Connection Connect</td>
</tr>
<tr>
<td>dump</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>echo</td>
<td>Data / Write</td>
</tr>
<tr>
<td>enable_bridging</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>enable_connect_server</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>enable_dde_client</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>enable_dde_server</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>enable_scripting</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>error</td>
<td>none</td>
</tr>
<tr>
<td>exception_buffer</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>execute_plugin</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>exit</td>
<td>Connection Shutdown</td>
</tr>
<tr>
<td>failed_license</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>flush</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>force</td>
<td>Data / Write, Force</td>
</tr>
<tr>
<td>format</td>
<td>Connection / Connect</td>
</tr>
<tr>
<td>heartbeat</td>
<td>none</td>
</tr>
<tr>
<td>ignore</td>
<td>Data / Read</td>
</tr>
<tr>
<td>ignore_old_data</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>include</td>
<td>Connection LoadConfig</td>
</tr>
<tr>
<td>instance</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>load_config_files</td>
<td>Admin / LoadConfig</td>
</tr>
<tr>
<td>load_plugin</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>Command Name</td>
<td>Permissions Required</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>load_scripts</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>lock</td>
<td>Data / Write</td>
</tr>
<tr>
<td>log_file</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>log_to_file</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>master_host</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>master_service</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>mirror_master</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>mirror_master_2</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>mult</td>
<td>Data / Write</td>
</tr>
<tr>
<td>on_change</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>point</td>
<td>Data / Write</td>
</tr>
<tr>
<td>private_attribute</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>property</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>quality</td>
<td>Data / Write</td>
</tr>
<tr>
<td>read</td>
<td>Data / Read</td>
</tr>
<tr>
<td>readid</td>
<td>Data / Read</td>
</tr>
<tr>
<td>register_datahub</td>
<td>Data / Read</td>
</tr>
<tr>
<td>report</td>
<td>Data / Read</td>
</tr>
<tr>
<td>report_all</td>
<td>Data / Read</td>
</tr>
<tr>
<td>report_domain</td>
<td>Data / Read</td>
</tr>
<tr>
<td>report_errors</td>
<td>Data / Read</td>
</tr>
<tr>
<td>request</td>
<td>Data / Read</td>
</tr>
<tr>
<td>request_initial_data</td>
<td>Data / Read</td>
</tr>
<tr>
<td>secure</td>
<td>Data / Write</td>
</tr>
<tr>
<td>set</td>
<td>Data / Write</td>
</tr>
<tr>
<td>show_data</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>show_debug_messages</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>show_event_log</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>show_icon</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>show_properties</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>show_script_log</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>slave</td>
<td>Data / Read</td>
</tr>
<tr>
<td>subassembly</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td><strong>Command Name</strong></td>
<td><strong>Permissions Required</strong></td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>success</td>
<td>none</td>
</tr>
<tr>
<td>sync</td>
<td>Data / Write</td>
</tr>
<tr>
<td>taskdied</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>taskstarted</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>tcp_service</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>timeout</td>
<td>none</td>
</tr>
<tr>
<td>transmit_insignificant</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>type</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>unload_plugin</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>unreport</td>
<td>Data / Read</td>
</tr>
<tr>
<td>version</td>
<td>none</td>
</tr>
<tr>
<td>warn_of_license_expiry</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>write</td>
<td>Data / Write</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>OPC-specific commands</strong></th>
<th><strong>Permissions Required</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_opc_client</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>enable_opc_server</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>OPCAddItem</td>
<td>Data / Write</td>
</tr>
<tr>
<td>OPCAttach</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>OPCDetach</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>OPCInit</td>
<td>Admin / ConfigAdmin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>DDE-specific commands</strong></th>
<th><strong>Permissions Required</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>DDEAdvise</td>
<td>Data / Write</td>
</tr>
<tr>
<td>DDEConnect</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>DEInit</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>DDEService</td>
<td>Admin / ConfigAdmin</td>
</tr>
<tr>
<td>DDEUnadvise</td>
<td>Data / Write</td>
</tr>
<tr>
<td>DDEUnadvisePoint</td>
<td>Data / Write</td>
</tr>
<tr>
<td>EnableDDEServer</td>
<td>Admin / ConfigAdmin</td>
</tr>
</tbody>
</table>

**Passwords**

The authentication information for passwords is stored in a database in the configuration directory in a non-reversible encryption. They are secure and non-recoverable. If a user forgets his password, it cannot be retrieved or regenerated.
When a password is associated with a mirror/tunnel connection, it is stored in a weakly encrypted form on disk, in the `Cogent DataHub.cfg` file. This is a reversible encryption, so a good security policy would be to deny access to this file to untrusted users.

When a password is transmitted across the network, it is transmitted in plain text. This is necessary to accommodate the variety of clients that could generate an authentication request. If the network is itself insecure, it is advisable to use a VPN (Virtual Private Network) or enable SSL for mirror/tunnelling to encrypt the network traffic.
Email and SMS

Introduction

The Cogent DataHub lets you send emails and SMS text messages, triggered by a DataHub event such as a point value change, or by a timer. The emails and messages can be in plain text or HTML format, and they can contain current values for any data point in the DataHub.

With this feature of the DataHub you can:

• Send SMS text messages to cell phones when an alarm event is triggered.
• Design end-of-day reports that are delivered to managers’ email accounts each morning.
• Provide managers with regular email updates of production targets.
• Emails can contain data from OPC servers, ODBC databases and other sources.
• Eliminate errors associated with manually writing production reports.
• Have the DataHub collect vital report information, format it as an Excel spreadsheet and then email the file to key people for review.

How it works

The DataHub has a built-in mailing program. DataHub scripts tell the program what messages to send, to whom, and when to send them. A typical mailing script contains instructions to send an email in plain text, HTML format using ASP, or both. The example script, MailTest.g contains examples of both methods. You can run this script to test the mailer, and then use the examples that follow to send your own messages.

The sections in this chapter show you how to:

1. Configure the mailer.
2. Send a test email message.
3. Send your own email messages.
4. Send SMS messages.
5. Create HTML email messages.
Configuring the Mail Server

Before you can send email from the Cogent DataHub, you will need to configure the DataHub mail server program, as follows:

1. With the DataHub running, right click on the DataHub system-tray icon and choose Properties.
2. In the Properties window, select Email/SMS.

Enter the information that you want to use for sending the email. This can be the same as the SMTP server listed in your email client program.

SMTP Server:
The name of the SMTP server.

Port:
The SMTP port number (typically this is port 25).

Sender Email:
The email address of the sender. This will appear in the From field of the email. The address can be in either of these two forms:

- username@datadomain.com will be displayed as username@datadomain.com in the email reader (client).
- User Name <username@datadomain.com> will be displayed as User Name in the email reader (client).

User name:
The log-in name you use to access this SMTP account.

Password:
The applicable password.

You will need to know your email account user name and password.
in order to have the DataHub successfully send to the outgoing SMTP Server. If you have problems, then look in the DataHub Script Log to view any error messages. Typically, problems are due to incorrect user name and password.

3. In the **Security** section:

Choose one of the three SSL options, and specify whether you want to accept invalid or untrusted security certificates.

4. Click the **Apply** or **OK** button to submit your entries.

The DataHub mailer is now ready to use. If you haven't already done so, we suggest sending a test message as explained in the next section.

**Sending a Test Message**

Once you have configured the mail server you can configure and send a test email. Here's how:

**Open the Email/SMS Events window**

1. In the Cogent DataHub Properties window, select **Email/SMS**.
2. In the **Configure Email and SMS Events** section press the **Configure** button.

This opens the Email/SMS Events window:
Define the Email Message

1. Select the **1. Email** tab.
2. For the **Message Type**, choose **Plain Text Message**.
3. Enter a recipient email address in the **Recipients** box. You can enter several addresses, separated by commas.
4. Enter a subject in the **Subject** box.
5. For the **Body**, choose **Use the following text**:
6. Start the DataSim program if it isn’t already running, and ensure that it is connected to the DataHub.
7. In the point-picker list on the right, expand the **DataSim** data domain and select the point named **Sine**.
8. Click the **Name** button to the right of the point-picker list.
9. In the text entry field, type the following:

   The point: DataSim:Sine

10. Click the **Insert Point** button. Your text display should now look like this:

    The point DataSim:Sine

11. Press **Enter** and continue typing:

    The point DataSim:Sine had a value of:

12. Click the **Value** button and then click the **Insert Point** button. Your text display
should now look like this:

| The point DataSim:Sine had a value of: <%=DataSim:Sine%> |

13. Press Enter and continue typing:

| The point DataSim:Sine had a value of: <%=DataSim:Sine%> at the time: |

14. Click the Time button and then click the Insert Point button. Your text display should now look like this:

| The point DataSim:Sine had a value of: <%=DataSim:Sine%> at the time: <%=PointTimeString(DataSim:Sine)%> |

The message is ready. Now you can assign a trigger and set a condition.

**Assign a Trigger**

For this example, we will trigger the action on the DataSim:UpdateFrequency point.

1. Select the 2. Trigger tab.
2. From the point selector, expand the DataSim data domain and select the point UpdateFrequency.
3. Click the + button to the right of the Point Name field. The point name DataSim:UpdateFrequency should fill in for you.

You can choose any point for the trigger, including the point that gets written, such as DataSim:Sine in our example. For more information about triggers, please refer to the section called “Assigning a Trigger”.

**Set a Condition and Configure the Action**

For this example, let's limit the trigger on the DataSim:UpdateFrequency point to changes only to values over 100.

1. Select the 3. Condition tab.
2. Click the checkbox in the first row.

3. From the point selector, expand the DataSim data domain and select the point UpdateFrequency.

4. Click the + button in the left column. The text $DataSim:UpdateFrequency should fill in the box.

5. From the drop-down box, choose the > operator.

6. In the right column, enter the number 100. Your screen should now look like this:

![Screen shot of condition setup]

You have set the condition. The expression at the bottom shows what will be passed to Gamma, the internal scripting engine of the DataHub.

7. Go down to the Configured Actions box and click the Create button.

![Screen shot of configured actions]

A new configured action should appear in the list. This is a summary of what you have done. When a configured action is selected in this list, you can make changes in any of the tabs and modify it using the Modify. You can also duplicate a configured action using the Create button, or remove it with the Remove button. For more information about configured actions, please refer to the section called “Configured Actions”

8. Click the Apply button to activate the configured action. Now let's see how it all works.

Trigger and Send an Email

The action you just configured causes the DataHub to send an email any time the DataSim Update Frequency is changed to a value greater than 100. To test the script, you'll need to trigger it by changing that value in the DataSim.
1. In DataSim, press the **More...** button to view the **Configurable Options**
2. Change the **Update Frequency** to a number greater than **100** and click the **Apply Changes** button to commit the change.

![Configurable Options](image)

3. Check the email account of the recipient. You should have received a message that looks like this:

   ![Email Message](image)

   Each time you enter a new **Update Frequency** value greater than **100** in the DataSim, the DataHub script will send a similar message.

You have just configured and tested an action that sends an email with the name, value, and timestamp of the *Sine* point in the *DataSim* data domain whenever the value of the *UpdateFrequency* point changes to a value over **100**. Now you can configure other emails to send your own text messages or HTML pages. The remaining sections in this chapter explain the interface in more detail.

**Defining the Email Message**

To send an email you need to determine the type of email message, its recipients, title, and message body. This is done from the *Email/SMS Events* window, which you can access in this way:

1. In the Cogent DataHub Properties window, select **Email/SMS**.
2. In the **Configure Email** section press the **Configure** button.

![Configure Email and SMS Events](image)

3. Select the **1. Email** tab and configure your email as explained below.
Email and SMS

Message Type

- **Plain Text Message** sends the text of the message as written in the source file or entered in this interface. Data point values will be assigned at the time the message is sent.

- **HTML Message** sends the source file or entry in this interface as an HTML file. Data point values will be assigned at the time the message is sent.

Recipients

This can be a single email address, or a list of email addresses where each address is separated by a comma. Addresses can be in either of these two forms:

- username@datadomain.com
- User Name <username@datadomain.com>

It is also possible to create a dynamic list of recipients, as explained in the section called “Dynamically Changing Email Subjects and Recipients”.

Message Content

- **Subject** Enter the subject of the message.
- **Body** You can use a message from a file, or compose one in the editing box.
- **Use this file**: lets you insert the name of a file that you want to send as the text of your email. This is not an attachment, but rather the body of your message. Press the
... button to browse for the file you need. To see some HTML file examples, please refer to the section called “HTML Message Examples”.

- **Use the following text:** lets you write and edit the body of your message. To insert the name, value, timestamp, or quality of the point in the point-picker list, select **Name, Value, Time, or Quality** button as desired. Then click the **Insert Point** button. The DataHub will insert into your text the point name with the proper syntax for the desired output in the email.

If you want to send a message from a file, you can still use the text editor with its convenient interface to create it. Write up the message in the editor, then copy and save it to a file.

The value, time, and quality attributes of the DataHub points are accessed by using a special syntax. This is applied automatically in the text editor when you press the **Insert** button. For your reference, the syntax is as follows:

<table>
<thead>
<tr>
<th>Button</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>$domainname:pointname</td>
<td>DataSim:Sine</td>
</tr>
<tr>
<td>Value</td>
<td>&lt;%= $domainname:pointname %&gt;</td>
<td>&lt;%= $DataSim:Sine %&gt;</td>
</tr>
<tr>
<td>Time</td>
<td>&lt;%= PointTimeString (#$domainname:pointname) %&gt;</td>
<td>&lt;%= PointTimeString (#DataSim:Sine) %&gt;</td>
</tr>
<tr>
<td>Quality</td>
<td>&lt;%= PointQualityString (#$domainname:pointname) %&gt;</td>
<td>&lt;%= PointQualityString (#DataSim:Sine) %&gt;</td>
</tr>
</tbody>
</table>

In this syntax, the special characters are used as follows:

<table>
<thead>
<tr>
<th>Character</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;% ... %&gt;</td>
<td>The enclosed expression will be evaluated by Gamma, the DataHub scripting language.</td>
</tr>
<tr>
<td>$</td>
<td>Indicates to Gamma that this is a DataHub point name.</td>
</tr>
<tr>
<td>PointTimeString()</td>
<td>A Gamma function that returns the timestamp of a DataHub point in an easily readable format.</td>
</tr>
<tr>
<td>PointQualityString()</td>
<td>A Gamma function that returns the quality of a DataHub point, as a text string.</td>
</tr>
<tr>
<td>#</td>
<td>Protects the DataHub point from being evaluated by Gamma until the function is called.</td>
</tr>
</tbody>
</table>

**Assigning a Trigger**

A trigger is an event that causes the email to be sent. A trigger event can be either a point value change, a timer event, or a calendar event. You can assign a different trigger for each email, or an identical trigger to any number of emails. An action can be configured to execute on every trigger event, or you can assign **trigger conditions** that are evaluated whenever a trigger occurs, to determine if the action should be executed.
The three kinds of triggers are:

- **Point Change** fires whenever a specified trigger point changes.
  
  1. Type the name of the point into the **Point Name** box, or select the point using the data tree on the right, then click the + button.
  
  2. (Optional) Enter a value deadband if you want to filter out extraneous data. The number you enter will specify a high and low (plus or minus) range. Any value change falling within that range will not cause the trigger to fire. A positive or negative change greater than this value will activate the trigger and cause the email to be sent.

  To create a trigger that gets reset automatically, please refer to An Auto-Resetting Trigger in the section called “Setting Trigger Conditions”.

- **Repeat Timer** fires cyclically, each time the number of seconds elapses.

- **Time of Day** fires at the time you specify. You can enter:
  
  - A number, indicating a specific value. For example, a 0 in the seconds field would cause the event only on the 0th second of the minute. A 30 would indicate only on the 30th second of the minute.
  
  - A list of numbers, separated by commas. For example, entering 0, 15, 30, 45 in the minutes field would indicate that the event should fire on the hour and at 15, 30 and 45 minutes past the hour.
  
  - A range of numbers, separated by a dash. For example, entering 8–18 in the hours field would indicate that the event should fire every hour from 8 a.m. to 6 p.m.. Ranges can be mingled with lists, as in 0, 4, 8–16, 20.
  
  - An asterisk (*) indicates that the event should fire for every possible value of the field. For example, a * in the seconds field would cause the event to fire every second. A * in the hours field would cause the event to fire every hour.

  To regularly log a record on specific days of the week, please refer to the
section called “Setting Trigger Conditions”.

The ranges of the fields are:

- **Year**: 1970-*
- **Month**: 1–12
- **Day**: 1–31
- **Hour**: 0–23
- **Minute**: 0–59
- **Second**: 0–59

The year and month are entered differently here than for the Gamma localtime function, as explained in Time Conditions.

Examples:
- These entries:

  ![Example](image)

  would cause an email to be sent at 8:45 every day, every month, and every year.

- These entries:

  ![Example](image)

  would cause an email to be sent every hour on the 15th day of each month, every year.

- These entries:

  ![Example](image)

  would cause an email to be sent every second for 5 minutes, every two hours between 8 a.m. and 6 p.m.

**Setting Trigger Conditions**

Each action can have up to four conditions that determine whether an email gets sent when the trigger fires.
Fill in the conditions according to the guidelines below. Check the box next to the condition to apply it. As you make entries, the corresponding Gamma code will appear in the display. Gamma is the DataHub's built-in scripting language. The code that appears in the Expression box is the actual code that gets run by Gamma. The order of precedence for "And" and "Or" operators (&& and ||) is first And, then Or.

**Point Value Conditions**

Point names can be entered on either or both sides of the comparison. They can be picked from the data tree list, or typed in. Each point name needs to have a dollar sign ($) in front of it to indicate to Gamma that this is a DataHub point. You can put numerical values into either side of the comparison.

When you enter a point name in a condition field, the current value of the point will be used in the evaluation. For example, you could define a condition that states that whenever the trigger event occurs, the action will only be executed if another point value is within a certain range.

There are three automatic variables available for working with point values:

- lasttrigger - the value of the trigger point the last time this trigger was fired.
- thistrigger - the value of the trigger point now.
- lastevent - the value of the trigger the last time the event was actually executed.

**Time Conditions**

This provides an additional way to restrict the time, day, month, etc. when a message gets sent. In addition to the options on the triggers, here you have day-of-week condition statements which can give you more flexibility for events based on specific days of the week. These will work with any type of trigger event.

You can use the Gamma functions clock and localtime to specify particular days of the week. For example, these entries:
Email and SMS

would create this Gamma code:

```
(localtime(clock()).wday > 0 && localtime(clock()).wday < 6)
```

which would cause an email to be sent only Monday through Friday. The function `localtime` returns a class whose members contain information about the date, as follows:

- `.sec` The number of seconds after the minute (0 - 59).
- `.min` The number of minutes after the hour (0 - 59).
- `.hour` The number of hours past midnight (0 - 23).
- `.mday` The day of the month (1 - 31).
- `.mon` The number of months since January (0 - 11)
- `.year` The number of years since 1900.
- `.wday` The number of days since Sunday (0 - 6).
- `.yday` The number of days since January 1 (0 - 365)
- `.isdst` 1 if daylight saving time is in effect, 0 if not, and a negative number if the information is not available.

The year and month are entered differently here than for Time of Day trigger conditions, as explained in the section called “Assigning a Trigger”.

There are two automatic variables available for working with time values:

- `lasteventtime` - the time that the last event was executed, in UNIX epoch time.
- `curtime` - the UNIX epoch time now.

Custom Conditions

If the conditions you need to meet are beyond the scope of this interface, you can use a Gamma function to express virtually any condition you need. Then you can insert the function into one of the condition boxes, and set a condition based on the return value of the function.

To do this is you can create a DataHub script (.g file) that contains only the functions you will be using for conditions, without any classes or methods. For example, here is the complete contents of such a file, named `MyConditions.g`:

```plaintext
function MyFunction ()
{

```
myvalue = $DataSim:Sine;
princ("Value when the trigger fired: ", myvalue, "\n");
myvalue;
}

This function prints the value of the $DataSim:Sine point, and returns its value. We can use this function as a condition by calling it from one of the condition boxes in the interface, like this:

When the trigger fires, MyFunction is called, and the return value gets checked to see if it is less than .4. If so, the email is sent.

An Auto-Resetting Trigger

This script can turn any DataHub point into a trigger that automatically resets. To use it, you first need to load and run the TriggerFunctions.g script (shown below and included in the installation archive). Then, if you put this formula:

`HighWithReset($default:TriggerPoint)!=$default:TriggerPoint` != nil

into the condition boxes, whenever the `TriggerPoint` changes to a non-zero number in the DataHub, your trigger will fire. The script waits for a millisecond, then resets the `TriggerPoint` back to zero. The second function works similarly, but triggers on a change to zero, instead of a change to a non-zero number.

TriggerFunctions.g

/*
 * This file contains handy functions to perform more complex
 * condition handling in the Condition tab of the data logging
 * and email interfaces.
 */

/*
 * Test a trigger point for a non-zero value. If the point is
 * non-zero, create a delayed event to reset the point to zero,
 * and return true, indicating that the condition has succeeded
 * and the action should proceed. If the value is 0, then simply
 * return nil indicating that the action should not proceed. We
 * need to test for zero because when we reset the trigger point
 * to zero a second data change event will occur.
 */
function HighWithReset(!triggerpoint)
{
    local value;
    if (!undefined_p(value = eval(triggerpoint)) && value != 0)
    {
        after(0.001, `setq(@triggerpoint, 0));
        t;
    }
    else
    {
        nil;
    }
}

function LowWithSet(!triggerpoint)
{
    local value;
    if (!undefined_p(value = eval(triggerpoint)) && value == 0)
    {
        after(0.001, `setq(@triggerpoint, 1));
        t;
    }
    else
    {
        nil;
    }
}

Configured Actions

A configured action will cause a given email to be sent, based on a trigger and optional conditions. It is the end result of your configuration activities in this interface. The Configured Actions list shows the actions you have configured, and allows you to create, modify, or remove actions, as well as turn them on or off.
The list of configured actions shows the actions you have already configured. Selecting an existing action from the list automatically fills in the Email, Trigger, and Condition tabs with its information. Checking or unchecking the On box at the left lets you switch the action on or off.

The Create button creates an action for the information currently entered in the Email, Trigger, and Condition tabs. If you press the Create button while a configured action is selected, it creates a duplicate of that configured action and adds it to the list. This is a quick way to configure similar actions.

The Modify button overwrites the selected configured action with the information currently entered in the Email, Trigger, and Condition tabs.

The Remove button removes a configured action.

Once a configured action has been created or modified, the changes won't take effect until you click the Apply or Done button.

Sending SMS Text Messages

Sending SMS text messages from the DataHub is the same as sending an email, but it must be a plain text email of 140 characters or less, sent to the appropriate SMS gateway email address.

Most cell phone service providers offer email and text messaging options on new subscriptions. For example, if you have a cell phone subscription with Telus in Canada, and your cell phone number is (416) 123 4567, then the SMS gateway email address for this phone would be:

4161234567@msg.telus.ca

Normally, SMS text messages are sent from one cell phone to another, which is all handled within the cell network itself. Using the SMS text message address, you can send a plain text email and it will be converted by the SMS gateway into a text message and be delivered to your phone. There may be a short delay while the conversion from email to text message occurs, but the message usually arrives in less than a minute.

Check with your cell phone service provider for the SMS text message address for your cell phone number. If your cell phone plan does not include an Email to SMS gateway ser-
vice, then an Internet search will provide a wide range of third-party companies offering inexpensive SMS gateway services.

**HTML Message Examples**

Sending an HTML message is as simple as clicking the **HTML Message** button in the **Email** tab. For the **Message Source** you can choose a file or write HTML code directly into the text-entry box. Here are two examples of how you can embed data into an HTML messages, using an ASP source file

**An HTML Message with Embedded Data Points**

This is an example of an ASP file that embeds the latest data from DataHub points into an HTML table. The ASP file is named **EmbedPoints.asp**, and comes installed with the DataHub in the **Cogent DataHub/etc/** directory. If the DataHub is configured to send this file as the message body, and the DataSim’s UpdateFrequency is changed to, say, 102, the DataHub will email a message like this:

![Warning: The DataSim UpdateFrequency has been set to greater than 100Hz.](image)

### Contents of the ASP File

The contents of the file **EmbedPoints.asp** are as follows:

```html
<html>
<style>
BODY, P, TD{
    background-color : White;
    font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size : 8pt;
}
TH{
    font-family:Verdana, Arial, Helvetica, sans-serif;
    font-size: 9pt;
    font-weight: bold;
```

```
```

```
This is a simple example of an HTML template file which contains embedded point values from the DataHub.

Current DataSim status is

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSim:Sine</td>
<td>$DataSim:Sine%</td>
</tr>
<tr>
<td>DataSim:Ramp</td>
<td>$DataSim:Ramp%</td>
</tr>
<tr>
<td>DataSim:Square</td>
<td>$DataSim:Square%</td>
</tr>
<tr>
<td>DataSim:Triangle</td>
<td>$DataSim:Triangle%</td>
</tr>
<tr>
<td>DataSim:UpdateFrequency</td>
<td>$DataSim:UpdateFrequency%</td>
</tr>
<tr>
<td>DataSim:Amplitude</td>
<td>$DataSim:Amplitude%</td>
</tr>
</tbody>
</table>
This file consists of HTML code interspersed with Gamma code. Gamma is the scripting language of the Cogent DataHub. The Gamma code is often used to determine the value of a DataHub point, with the following syntax:

`<%@domainname:pointname%>`

The pointed brackets and percent signs ( `<% ... %>` ) indicate to the DataHub ASP interpreter that this is Gamma code. The equals sign (=) tells Gamma to evaluate the expression, and the dollar sign ($) tells Gamma that this is a DataHub point.

**An HTML Message with a Table Created in Code**

This is an example of an HTML message with a table created by using code, rather than explicitly writing it out. Using code provides more flexibility in formatting the data and making changes to the table. The code is written into an ASP file named CreateTable.asp, which comes installed with the DataHub in the Cogent DataHub/etc/ directory. If the DataHub is configured to send this file as the message body, and the DataSim's UpdateFrequency is changed to, say, 102, the DataHub will email this message:

```html
<br><br>
<!-- Update frequency is too high

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Quality</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSim:Sine</td>
<td>-0.4398</td>
<td>Good</td>
<td>Mon Apr 23 12:57:29.175</td>
</tr>
<tr>
<td>DataSim:Ramp</td>
<td>0.2324</td>
<td>Good</td>
<td>Mon Apr 23 12:57:29.175</td>
</tr>
<tr>
<td>DataSim:Square</td>
<td>-0.5000</td>
<td>Good</td>
<td>Mon Apr 23 12:57:29.175</td>
</tr>
<tr>
<td>DataSim:Triangle</td>
<td>-0.3353</td>
<td>Good</td>
<td>Mon Apr 23 12:57:29.175</td>
</tr>
<tr>
<td>DataSim:UpdateFrequency</td>
<td>102.0000</td>
<td>Good</td>
<td>Mon Apr 23 12:57:28.865</td>
</tr>
<tr>
<td>DataSim:Amplitude</td>
<td>1.0000</td>
<td>Good</td>
<td>Mon Apr 23 12:57:28.865</td>
</tr>
<tr>
<td>DataSim:Frequency</td>
<td>0.1000</td>
<td>Good</td>
<td>Mon Apr 23 12:57:28.865</td>
</tr>
<tr>
<td>DataSim:Offset</td>
<td>0.0000</td>
<td>Good</td>
<td>Mon Apr 23 12:57:28.865</td>
</tr>
</tbody>
</table>

Warning: The DataSim UpdateFrequency has been set to greater than 100Hz.
Current DataSim status is... -->
```
Contents of the ASP File

The contents of the file `CreateTable.asp` are as follows:

```html
<html>
<style>
BODY, P, TD{
    background-color : White;
    font-family : Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size : 8pt;
}
TH{
    font-family:Verdana, Arial, Helvetica, sans-serif;
    font-size: 9pt;
    font-weight: bold;
    background-color: #cce6fe;
}
.highlight{background-color:#FFFFCC; text-align:right}
.warning{color: #FF0000; font-weight: bold;}
</style>
<body>
<p></p>
<div class="warning">Warning: The DataSim UpdateFrequency has been set to greater than 100Hz.</div>
<p></p>
Current DataSim status is
<p></p>
<table border="1">
  <tr>
    <th width="180">Name</th><th width="80">Value</th><th width="80">Quality</th><th width="160">Timestamp</th>
  </tr>
  <% require ("Time"); require ("Quality");
  try
  {
    local v, q, tm, ts, info;
    with pt in [ #$DataSim:Sine, #$DataSim:Ramp, #$DataSim:Square, #$DataSim:Triangle, #$DataSim:UpdateFrequency, #$DataSim:Amplitude, #$DataSim:Frequency, #$DataSim:Offset ] do
      info = PointMetadata (pt);
  }
```
Email and SMS

This file consists of HTML code interspersed with Gamma code. Gamma is the scripting language of the Cogent DataHub. The Gamma code is often used to determine the value of a DataHub point, with the following syntax:

```
<%= $domainname:pointname %>
```

The pointed brackets and percent signs (`<% ... %>` indicate to the DataHub that this is Gamma code. The equals sign (`=`) tells Gamma to evaluate the expression, and the dollar sign (`$`) tells Gamma that this is a DataHub point. Other Gamma statements and functions used in this example include `require`, `try`, `local`, `with`, `if`, `format`, `catch`, `princ`, and `print_stack`. The functions `GetQualityName` and `PointGetUnixTime` are from the required files `Quality.g` and `Time.g` respectively.

**Dynamically Changing Email Subjects and Recipients**

The ASP processor in Gamma allows you to embed the result of any Gamma expression within the subject and recipient fields of an email. To do this on the subject field, you would use the same `<%= %>` syntax as is available for messages, for example:

```
The Sine value is now <%= $DataSim:Sine %>
```

would put the value of the `DataSim:Sine` point into the subject line of the email or message.
This syntax, explained in the section called “Defining the Email Message” can also be used to insert addresses for one or more the message recipients, by creating a point that contains the list of recipient names. The value of this point could then be changed externally based on who is on-call or is logged into an attached SCADA system. For example, a point in the default domain named CurrentOperatorEmail, would be entered in the Recipients: field like this:

```<%= $default:CurrentOperatorEmail %>```

If you need a more complex calculation to determine the recipients, you can create a Gamma script that loads when the DataHub starts. For example, to change the email based on the value of a point, you could do something like this:

```gamma
function choose_mail_recipient()
{
    if ($DataSim:Sine > 0.5)
        "operator1@gmail.com";
    else
        "operator2@gmail.com";
}
```

and then put the appropriate function call into the email recipient list, like this:

```<%= choose_mail_recipient() %>`

Notice that the expression within `<%=` does not end with a semicolon. This syntax requires a Gamma *expression*, not a Gamma *statement*. Effectively, it needs to be code that would be syntactically correct in this statement:

```gamma
x = insert_expression_here;
```

You can add as many function statements to your script as you like. Don’t use method statements for this, since they are just for scope of the class of that script. Once a function has been defined in a running Gamma script, it is available to all other running Gamma programs. If you have created other Gamma programs, put this one at the top of the list, so that the function becomes available before those programs start. The Email/SMS program starts after the programs in the list.
System Monitor

Introduction

The Cogent DataHub System Monitor provides a way to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. in the Cogent DataHub.

For example, by monitoring process ID you could determine whether a particular process is running or not. Any information accessed here becomes part of the DataHub’s data set, and can thus be tunneled across the network, used in scripts or as email triggers, viewed in a spreadsheet, or stored in a database.

Configuring the System Monitor

1. With the Cogent DataHub running, right click on the DataHub system-tray icon and choose Properties.

2. In the Properties window, select System Monitor.
To enable system monitoring, check the **Enable system monitoring** box and edit the configuration options as desired:

**Data Domain:**

The name of any DataHub data domain. The values retrieved from the system will be shown as points in this data domain.

**Update Rate:**

The frequency that the system is polled and all selected points are updated. The minimum polling time is 100 ms., so the value entered here cannot be less than 100.

A high update rate (a low number here) for many data points could use a great deal of CPU.

**When a parameter is invalid:**

A parameter will be invalid if the object being monitored is not available. For example, if a process is not running then the parameters for that process will all be invalid. This is a useful way to monitor a system process or other object. For example, you could use a script or other client to watch a process ID, and when the process ID becomes -1 you could generate an alarm indicating that the process is no longer running.

**Point Names:**

The System Monitor automatically creates Cogent DataHub point names based on the names of the system properties. Some client programs cannot work with point names containing special characters. This section allows you to specify which characters will be removed from the property name when constructing the
point name.

Now you are ready to create the list of system parameters that you want to monitor.

3. Click the **Select From List** button. This will open the Select System Properties dialog:

   ![Select System Properties Dialog](image)

Depending on your system, this dialog may take a few seconds to appear. If it does not come up, the Event Log will contain a message. Otherwise, just be patient, it will open eventually.

In the Select System Properties dialog you can specify which items to add to your list of monitored system properties, according to these criteria:

- **Performance object** A list of all available objects, such as CPU, Memory, Process,
Print Queue, TCP, etc.

- **Counters** All of the available data categories related to the selected performance object. You can choose all counters, or select specific counters from the list. The *Explain* button opens a window with an explanation of the selected counter.

- **Instances** All of the instances of the chosen performance object. For example, if you chose Process for your performance, this list will show all of the processes running on your system. You can choose all processes or select specific processes from the list.

  A number in this list normally indicates a selection from multiple objects of a given type, and _Total means the total across all of the objects. For example, if you are looking at Processor in a multi-processor machine, you will see a number (0, 1, etc.) for each processor and a _Total for the cumulative statistic over all processors.

4. Select a performance object, and counters and instances as applicable. For example, to see the process ID for DataSim, first ensure that DataSim is running, then select:

   - **Performance object** Process
   - **Select counters from list** ID Process
   - **Select instances from list** DataSim

5. Click the *Add* button to add the selected items to the *Monitor system parameter* list in the DataHub Properties window.

6. Click the *Apply* or *OK* button in the Properties window when you are finished making your choices and filling the list, to apply your changes. You should be able to view the results in the Data Browser.

   If you change your mind on what points to monitor, you can change the list at any time. Any points you remove from the list will continue to exist in the DataHub until it is shut down and restarted. Please refer to the section called “Data Points” for more information on creating and deleting points.

**Monitoring Systems Across a Network**

You can monitor a system across by using DataHub *mirroring*. *Mirroring* is how two or more instances of the Cogent DataHub link over a network or the Internet to maintain identical data sets.

*Mirroring* is the same as *tunnelling*, as described in the section called “Tunnel/Mirror”.

For every mirroring connection, you must assign one DataHub to be the master, and the other to be the slave. This determines which side initiates communication. Once communication is established, the data is identical. Generally it is recommended that the
DataHub on the machine being monitored act as the master, while the machine that is collecting the monitoring data be the slave. In a hub-and-spoke arrangement, that DataHub could be the slave to multiple masters, to collect all the data in a single DataHub.

**Configure the DataHub as a tunnel/mirror master**

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **Tunnel/Mirror**.
3. In the **Tunnelling Master** section, you can configure plain-text or secure tunnelling. Ensure that at least one of these is checked. If you want to change any of the other defaults, please refer to the section called “Tunnel/Mirror” for more information.
   
   To optimize throughput, un-check the **Try to send data even if it is known to be superseded** option. This will allow the DataHub to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

4. To support incoming **WebSocket** connections from DataHub tunnelling clients, you will need to configure the tunnelling master DataHub's **Web Server**. For WebSocket connections, we recommend using SSL, on port 443.
5. Click **OK** to close the Properties window.

You are now ready to configure the slave DataHub.

**Configure the DataHub as a tunnel/mirror slave**

The slave DataHub behaves exactly like the master DataHub except that the slave establishes the tunnelling connection initially, and reestablishes it after a network break.

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **Tunnel/Mirror**.
3. Check the box **Act as a tunnelling/mirror slave to these masters.**

4. Click the **Add Master...** button to assign a master to this slave. The **Tunnel/Mirror Master Configuration** window will open:

5. Type in the following information:
   - **Primary Host** the name or IP address of the computer running the tunnelling master DataHub.
   - **Port** the port number or service name for this host. You should use default port number (4502) unless you have changed the entry in the master DataHub.
   - **Secondary Host** gives you the option to have an alternate host and service/port number. On startup or after a network break, the DataHub will search first for the primary host, then for the secondary host, alternating between primary and secondary until a connection is made. If no secondary host is specified, the connection will be attempted on the primary host only.

This feature is not recommended for implementing redundancy because it only checks for a TCP disconnect. The DataHub **Redundancy** feature, on the other hand, provides full-time TCP connections to both data sources, for instantaneous switchover when one source fails for any reason. There is no need to start up the OPC DA server and wait for it to configure its data set. You can also specify a preferred source,
and automatically switch back to that data source whenever it becomes available. By contrast, the primary and secondary host in the tunnel can act as a primitive form of redundancy, but will only switch on a connection failure at the TCP level, which is only one sort of failure that a real redundancy pair must consider.

- **Local data domain** The data domain in which you plan to receive data.
- **Remote data domain** the master DataHub data domain from which you plan to receive data. Point names will be mapped from the remote data domain (on the master DataHub) into the local data domain (on this DataHub), and vice versa.

Unless you have a good reason for making these different, we recommend using the same data domain name on both DataHubs for the sake of simplicity.

- **Remote user name** The user name for TCP security, established on the tunnelling master, using the DataHub Security option in the Properties window.
- **Remote password** The password for TCP security, established on the tunnelling master, using the DataHub Security option in the Properties window.
- **Secure (SSL)** lets you establish a secure connection using SSL tunnelling as long as the tunnelling master DataHub you are attempting to connect to has been configured for secure connections. The additional options allow for a connection to be made even if the security certificate is invalid, or the host name does not match. We don't recommend using these options unless absolutely necessary. For more about SSL, please refer to the section called “SSL and Firewalls”.

- **WebSocket** lets you connect via WebSocket. This option is applied for both primary and secondary hosts, and allows you to enter a **Proxy address**, a **Proxy port number**, a **username**, and a **password** as needed. When tunnelling through a proxy, HTTP uses normal HTTP proxy, and HTTPS uses HTTP CONNECT proxy. You can select the **Always use HTTP CONNECT** to use it for HTTP as well as HTTPS.

The WebSocket protocol requires a web server to act as an intermediary. So, for this option you will need to use the DataHub Web Server on the tunnelling master DataHub (as explained here).

There is a DataHub running on a Skkynet cloud server that you can connect to for testing. Here are the parameters you will need to enter for it:

- **Primary Host** demo.skkynet.com
- **Port** Will be set automatically by the system, 80 for WebSocket and 443 for Secure (SSL).
- **Local data domain** cloud
- **Remote data domain** DataPid
- **Remote user name** demo/guest
- **Remote password** guest
• **WebSocket** Must be selected.
• **Secure (SSL)** Optional.

6. You now have several options for the mirrored connection.

![Configure options](image)

a. **Data Flow Direction** lets you determine which way the data flows. The default is bi-directional data flow between slave and master, but you can effectively set up a read-only or write-only connection by choosing that respective option.

   To optimize throughput, check the **Read-only Receive data from the Master, but do not send** option. Only do this if you actually want a read-only connection. If you do not require read-write access, a read-only tunnel will be faster.

b. **When the connection is initiated** determines how the values from the points are assigned when the slave first connects to the master. There are three possibilities: the slave gets all values from the master, the slave sends all its values to the master, or the master and slave synchronize their data sets, point by point, according to the most recent value of each point (the default).

c. **When the connection is lost** determines where to display the data quality as "Not Connected"—on the master, on the slave, or neither.

   If you have configured **When the connection is initiated** as **Synchronize based on time stamp** (see above), then this option must be set to **Do not modify the data quality here or on the Master** to get correct data synchronization.

d. **Connection Properties** gives you these options.
• **Replace incoming timestamp**... lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.

• **Transmit point changes in binary** gives users of x86 CPUs a way to speed up the data transfer rate. Selecting this option can improve maximum throughput by up to 50%.

  For more information, please refer to the section called “Binary Mode Tunnel/Mirror (TCP) Connections”.

• **Target is an Embedded Toolkit server** allows this slave to connect to an Embedded Toolkit server rather than to another DataHub.

• **Heartbeat** sends a heartbeat message to the master every number of milliseconds specified here, to verify that the connection is up.

• **Timeout** specifies the timeout period for the heartbeat. If the slave DataHub doesn't receive a response from the master within this timeout, it drops the connection. You must set the timeout time at least twice the heartbeat time.

  To optimize this setting, please refer to the section called “Tunnel/Mirror (TCP) Heartbeat and Timeout”.

• **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

7. Click **OK** to close the **Tunnel/Mirror Master** window. The fields in the **Tunnelling Slave** table of the Properties Window should now be filled in.

8. Click the **Apply** button in the Properties Window. If the master DataHub is running, this DataHub should establish the tunnelling connection, and the **Status** should display **Connected**. You can view the data with the **Data Browser**, or view the connection with the **Connection Viewer**.

Open the Data Browser and select the data domain you requested to mirror. If the master Cogent DataHub has been correctly configured, you should now see all the master DataHub data for that data domain.
Excel Connections

You can use the Cogent DataHub to put data into Excel, and to write data from Excel back to the DataHub.

The following sections explain how to drag and drop live data into Excel, how to configure the Cogent DataHub to receive data, and how to use Excel macros for sending and receiving data between Excel and the DataHub.

Independently of the DataHub’s DDE feature, you can use Vine Add-in to exchange data in real time with Excel—locally or over a network.

Getting Data into Excel

Before starting, to see any results you will have to ensure that you have some kind of data being fed into the Cogent DataHub. If your system isn't set up for this yet, you can create a local data feed by following the steps outlined in the section called “Test with simulated data”.

There are two ways to get data into Excel from the Cogent DataHub: by setting up a **DDEAdvise** loop to receive data automatically, or by using a **DDERequest** command from a macro to read data. Deciding which to use depends on your situation. We suggest you become familiar with both. For more information about DDE and these commands, please refer to the section called “DDE Protocol” and Appendix G, *DDE Overview*.

Method 1 - Drag and Drop using DDEAdvise

The easiest way to get data into Excel is to drag and drop point names from the DataHub Data Browser directly into the Excel spreadsheet. This automatically sets up a **DDEAdvise** loop between Excel and the DataHub. **DDEAdvise** loops update automatically so you will always see the latest data in your spreadsheet.

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **DDE**.
3. Ensure that the box Act as a DDE server is checked, and that the name datahub appears in the DDE Service Name area. If not, click the Add... button and add the name datahub.

4. Click OK to close the Properties window.

5. Right click on the Cogent DataHub system-tray icon and choose View Data from the pop-up menu to open the Data Browser.

6. Ensure that the Drag & Drop Style at the bottom of the Data Browser is set to MS-Office (Excel/Word).

7. Open an Excel worksheet.

8. In the Data Browser, click on the label for a point and drag it into the Excel worksheet.
You should see the data update in the worksheet at the same rate it is updating in the Cogent DataHub.

You can select multiple points for drag and drop by using Shift-click or Ctrl-click.

You can drag and drop timestamps and other attributes of a point using the Property dropdown list. Please refer to Drag and Drop Style and Property in the Data Browser section for more details.

If your data displays but does not update, you might need to change your settings in Excel. Please refer to the section called “Basic Trouble-Shooting for Excel Connections” for more information.

When you save and close a spreadsheet connected to the Cogent DataHub, and then attempt to reopen it, you may get one or more messages, depending on your security settings in Excel, or other circumstances. Here’s a summary of each message, and what to do:

This document contains macros. Enable them?
Click Enable Macros.

This workbook contains links. Update them?
Click Update. If the DataHub is already running, all the links should then update automatically. If the DataHub is not running, you will get a #REF! entry in each cell that has an advise loop established with the DataHub, and the next message (see below) will probably appear.
Remote data not accessible. Start DataHub?

Click No. At this point the best thing to do is close the worksheet, start the DataHub manually, and then reopen the worksheet. When you update the spreadsheet (see above) this time you won’t get any #REF! entries. If, instead of No you click Yes at this point, the DataHub will not start, but instead generate an error message, and Excel may even crash later on.

Method 2 - Excel Macros using DDERequest

Sometimes, you may prefer to manually read data into your spreadsheet, rather than use a DDEAdvise loop to constantly accept new values. It may be that you intend to print reports only a couple of times a day and don’t need to see every point change in between. You can have Excel read specific data points from the Cogent DataHub at your request by triggering the DDERequest command from within a macro.

Using DDERequest within a macro gives you complete control over when Excel reads new point values, and lets you read several data points at one time. To run the macro, it is convenient to link it to a control button. This is explained in Add a Control Button.

Create a macro

1. Open a spreadsheet.
2. From the Tools menu, select Macro, and then Macros....
3. In the Macro Name: field of the Macro dialog box, type the name GetInput, and press the Create button.
4. In the Visual Basic text entry window that comes up, edit the macro to read as follows:

```vbnet
' GetInput Macro

Sub GetInput()
    mychannel = DDEInitiate("datahub", "default")
    Application.Worksheets("Sheet1").Activate
    newval = DDERequest(mychannel, "my_pointname")
    Sheet1.Cells(2, 3) = newval
    DDETerminate mychannel
End Sub
```

Use the name of your data point from the Cogent DataHub for my_pointname.

We use cell C2 in this example. If you need to use another cell, you will have to replace (2, 3) with the row and column numbers of the cell you wish to use.

5. Save and close the Visual Basic text entry window.
Add a Control Button

1. Activate the **Forms** toolbar by clicking on the **View** menu and selecting **Toolbars**, and then **Forms**.

![Form Toolbar]

2. Click on the button icon, and then click in cell **D2**. (We use this cell in our example, but you can choose another cell if you'd like.) An **Assign Macro** window should appear.

![Assign Macro Window]

3. Select **GetInput** and click **OK**.

4. Change the label on the button to "**Get**".

![Excel Grid with Button]

5. For appearance, you can move the button, resize it with the handles, and change the size of the text by right-clicking on it and selecting **Format Control**.

6. Save the spreadsheet.

Receive the data

1. Now you're ready to receive the data. Open the DataHub Data Browser if it is not already open, go to the **default** data domain, and find the name of the point.

2. Click on the point to highlight it. The point name should appear in the **Selected Point:** field at the top of the Data Browser.

3. Type a new value for the point into the **Enter new value:** field and press **Enter**.
4. Go to Excel and click the **Get** button. You should see the data update each time you click the button.

### Getting Data out of Excel

There are two ways to get data out of Excel and into the Cogent DataHub:

1. **Configure a DDEAdvise loop** in the DataHub that instructs Excel to send data automatically to the DataHub any time a value changes. The data is sent immediately to the DataHub, every time the specified cell or range changes. This does not allow any kind of sanity check or safeguard on the data being sent, but in some cases it may be desirable to have Excel emit data automatically.

   Each time data is sent for one point (data item), it is sent for all points. This can tie up your network if you have a large number of points. If you need to send data for a large number of cells, you can reduce this effect and reduce CPU load by sending a range that contains the cells.

2. **Write a macro in Excel** that uses the **DDEPoke** command to ‘push’ data from Excel to the DataHub. This allows you to define exactly when the data is sent to the DataHub.

   ![Excel Connections](image)

   ![Image of Excel Connections](image)

   ![Image of Excel Connections](image)

### Method 1 - Configuring DDEAdvise loops in the Cogent DataHub

The quickest and easiest method to get data from Excel to the DataHub is to configure one or more **DDEAdvise** loops in the DataHub to automatically receive data from Excel, which is acting as a DDE server.

1. Open an Excel spreadsheet.
2. Choose a cell or range to hold the data you want to put into the DataHub. You will need to refer to your cell or range by row and column number, or by a name. For example, the cell B2 can be referred to as B2C2, or by giving it a name.
To name a cell or range, select it and enter a unique name the box just above the first column of the worksheet. Then save the worksheet.

3. Start the Cogent DataHub if it isn’t already started, and open the Properties Window (by right-clicking on the DataHub icon in the Windows system tray and selecting Properties).

4. Click the DDE button.

5. Make sure the Act as DDE client box is checked.

   For best performance, ensure that a DDE server (in this case, Excel) is running when using the DataHub as a DDE client. A DDE client can consume substantial system resources trying to connect if a DDE server is not available.

6. Click the Add button. This opens the DDE Item Definition window where you can add Excel as a new DDE service.

7. Type in the following information:
   - **Connection Name** Choose a name to identify this connection. It must be unique among all DDE connections.
   - **Service** Type in Excel.
   - **Topic** Type the name of your worksheet file. In Windows XP, this name is the same
as what is shown after the dash in the title-bar of the Excel spreadsheet. More recent versions of Windows might not show the complete name in the title bar. In any case, you must use the complete file name, so if the worksheet is named "Book1", then your Topic is simply Book1, but if the worksheet is named Test.xls then your Topic needs to be Test.xls.

If you want to link to a cell or range which is not on the first sheet in the workbook, you need to put the filename in square brackets, followed by the sheet name. For example, if your worksheet name is Test.xls:

<table>
<thead>
<tr>
<th>Sheet in workbook</th>
<th>Service</th>
<th>Topic to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first sheet</td>
<td>Excel</td>
<td>Test.xls</td>
</tr>
<tr>
<td>An unnamed sheet (e.g. Sheet2)</td>
<td>Excel</td>
<td>[Test.xls]Sheet2</td>
</tr>
<tr>
<td>A named sheet (e.g. StockData)</td>
<td>Excel</td>
<td>[Test.xls]StockData</td>
</tr>
</tbody>
</table>

- **Item Names** Type in the row and column numbers or the name you entered as the cell or range name in Excel (in step 2 above).

8. Click the **Add** button. The fields **DDE Item**, **Point Name** and **Data Domain** are then added to the list of items associated with this **DDEAdvise** loop. You can continue to add points for other cells in your spreadsheet or click **OK** to close the dialog.

The **DDE Item** is associated with a point in the DataHub. You can change the **Point Name** and **Data Domain** to anything you want by double clicking on the name and typing a new name. When you click OK, the new point will be created in the DataHub.

9. Click **OK** to close the DDE Item Definition window. The new **DDEAdvise** loop is added to the list.

10. Click the **Apply** button for your changes to take effect. Once you have done this, you should see the **DDEAdvise** loop connection **Status** change to **Connected**.

11. Open the Data Browser by right clicking the DataHub icon in the system tray and selecting **View Data**

12. With the **default** data domain chosen, scroll down to see the name of the point.

13. In Excel, type a number into the cell or range you named in step 2, and press **Enter**. You should see the data update in the Data Browser.

Although this is an easy way to send data from Excel, it is not the most efficient when you have a large number of points to transmit. Whenever Excel transmits a data point using **DDEAdvise**, it also transmits the current value of every other point associated with any **DDEAdvise** loop.

Where you have a large number of cells to update, we have found it to be much more efficient to transmit the data as Excel **ranges**. In your **DDEAdvise** loop, define a range of cells that contains the data you want to transmit. Using Excel ranges will reduce the load on the computer and make it easier to configure your application.
Another option for reducing the load on the computer when transmitting a large number of points is to write an Excel macro that uses DDEPoke to transmit data on a timed basis, say once a second. Information on how to write a macro in Excel to do this is given below.

When you save and close a spreadsheet connected to the Cogent DataHub, and then attempt to reopen it, you may get one or more messages, depending on your security settings in Excel, or other circumstances. Here's a summary of each message, and what to do:

This document contains macros. Enable them?
Click **Enable Macros**.

This workbook contains links. Update them?
Click **Update**. If the DataHub is already running, all the links should then update automatically. If the DataHub is not running, you will get a #REF! entry in each cell that has an advise loop established with the DataHub, and the next message (see below) will probably appear.

Remote data not accessible. Start DataHub?
Click **No**. At this point the best thing to do is close the worksheet, start the DataHub manually, and then reopen the worksheet. When you update the spreadsheet (see above) this time you won't get any #REF! entries. If, instead of **No** you click **Yes** at this point, the DataHub will not start, but instead generate an error message, and Excel may even crash later on.

**Method 2 - Writing Excel macros that use the DDEPoke command**

Writing an Excel macro is perhaps the most flexible and efficient way to send data from Excel to the Cogent DataHub. By using the **DDEPoke** command in an Excel macro you have complete control over exactly when the data is transmitted. We will also explain how you can write an Excel macro to transmit multiple points at the same time (see **Additional Pointers** for more details).

In our example, we have chosen to 'add a control button' to run the macro, but you could also run your macro on a timed interval to produce an automatic update on a cycle that you control.

**Create a macro**

1. Open a spreadsheet.
2. From the **Tools** menu, select **Macro**, and then **Macros**..
3. In the **Macro Name:** field of the **Macro** dialog box, type the name **SendOutput**, and press the **Create** button.
4. In the Visual Basic text entry window that comes up, edit the macro to read as follows:

```
' SendOutput Macro
```

Sub SendOutput()
    mychannel = DDEInitiate("datahub", "default")
    Application.Worksheets("Sheet1").Activate
    Call DDEPoke(mychannel, "my_pointname", Cells(4, 3))
    DDETerminate mychannel
End Sub

Use the name of your data point from the Cogent DataHub for my_pointname.

We use cell C4 in this example. If you need to use another cell, you will have to replace (4, 3) with the row and column numbers of the cell you wish to use. You can also name a range to send multiple values as an array.

5. Save and close the Visual Basic text entry window.

Add a Control Button

This explanation is illustrated in the section called “Add a Control Button”. We repeat the text briefly here.

1. Activate the Forms toolbar by clicking on the View menu and selecting Toolbars, and then Forms.
2. Click on the button icon, and then click in cell D4. (You can choose another cell if you'd like.) An Assign Macro window should appear.
3. Select SendOutput and click OK.
4. Change the label on the button to "Send".
5. Save the spreadsheet.

Send the data

1. Now you're ready to send the data. Open the Cogent DataHub Data Browser if it is not already open, go to the default data domain, and find the name of the point.
2. In Excel, type a number in cell C4 (or the cell or range you assigned the macro to) and press Enter.
3. Click the Send button.
4. You should see the data update.

Additional Pointers

• To reduce CPU for large amounts of data, send arrays of data using ranges instead of sending the data for each cell as a separate point.
• If you are using Unicode characters in strings for DDEPoke commands, you should
check the **Accept non-English characters in Excel strings (slower)** button in the DDE option of the Properties window.

This will cause Excel to send your strings of Unicode characters correctly, although slower than numerical data.

- The **DDEInitiate** and **DDETerminate** commands that are used to open and close DDE links between applications are also very CPU expensive. When sending variables at frequent intervals it is more efficient to open a DDE channel at the beginning of the session and close it when you are finished. Here are two suggestions:

1. Send multiple points within a single set of **DDEInitiate** and **DDETerminate** commands. For example:

   ```vba
   Sub Cascade_Writeback_Many()
       mychannel = DDEInitiate("datahub", "default")
       Application.Worksheets("variables").Activate
       DDEPoke(mychannel, "pointname1", Cells(1,2))
       DDEPoke(mychannel, "pointname2", Cells(2,2))
       DDEPoke(mychannel, "pointname3", Cells(3,2))
       DDEPoke(mychannel, "pointname4", Cells(4,2))
       DDEPoke(mychannel, "pointname5", Cells(5,2))
       DDEPoke(mychannel, "pointname6", Cells(6,2))
       DDETerminate mychannel
   End Sub
   
   In this example the worksheet named **variables** contains six variables (**pointname1** through **pointname6**) that we wish to send to the Cogent DataHub. The **DDEInitiate** command opens the channel, then all six variables are sent to the DataHub before the link is closed.

2. Create a separate 'open' and 'close' macro for the worksheet, and place the **DDEInitiate** and **DDETerminate** commands in those macros. This will keep communication to the DataHub open for the whole time the worksheet is open. The only drawback is that your data transmission could get interrupted (see below).

   • If you need to send data continually from Excel to the Cogent DataHub you may run into problems using **DDEInitiate** and **DDEPoke**. When you open a DDE channel using the **DDEInitiate** statement, and follow it with several **DDEPoke** statements, there is a chance that the DDE channel may fail after some time. For this reason, if you need to keep a DDE channel open for an extended period of time, we suggest that you attempt to deal with DDE errors within the macro.

**Networking Excel**
You can use the Cogent DataHub to network Excel in real time, by using DataHub mirroring. Mirroring is how two or more instances of the Cogent DataHub link over a network or the Internet via DHTP to maintain identical data sets.

Mirroring is the same as tunnelling, as described in the section called “Tunnel/Mirror”.

Independently of the DataHub’s DDE feature, you can use Vine Add-in to exchange data in real time with Excel—locally or over a network.

To network Excel, on each node you need to connect Excel to the Cogent DataHub. Then a mirroring connection is configured between each DataHub. For every mirroring connection, you must assign one DataHub to be the master, and the other to be the slave. This determines which side initiates communication. Once communication is established, the data is identical. Generally it is recommended that the DataHub on the server or the machine least likely to shut down act as the master, while the slave be on the client machine. In a hub-and-spoke arrangement, that DataHub could be the slave to multiple masters, to collect all the data in a single DataHub.

### Configure the DataHub as a tunnel/mirror master
1. Right click on the Cogent DataHub system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror.
3. In the Tunnelling Master section, you can configure plain-text or secure tunnelling. Ensure that at least one of these is checked. If you want to change any of the other defaults, please refer to the section called “Tunnel/Mirror” for more information.

   To optimize throughput, un-check the **Try to send data even if it is known to be superseded** option. This will allow the DataHub to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

4. To support incoming WebSocket connections from DataHub tunnelling clients, you will need to configure the tunnelling master DataHub’s Web Server. For WebSocket connections, we recommend using SSL, on port 443.
5. Click **OK** to close the Properties window.

You are now ready to configure the slave DataHub.

**Configure the DataHub as a tunnel/mirror slave**

The slave DataHub behaves exactly like the master DataHub except that the slave establishes the tunnelling connection initially, and reestablishes it after a network break.

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **Tunnel/Mirror**.
3. Check the box **Act as a tunnelling/mirror slave to these masters**.
4. Click the **Add Master...** button to assign a master to this slave. The **Tunnel/Mirror Master Configuration** window will open:

5. Type in the following information:
   - **Primary Host** the name or IP address of the computer running the tunnelling master DataHub.
   - **Port** the port number or service name for this host. You should use default port number (4502) unless you have changed the entry in the master DataHub.
   - **Secondary Host** gives you the option to have an alternate host and service/port
number. On startup or after a network break, the DataHub will search first for the primary host, then for the secondary host, alternating between primary and secondary until a connection is made. If no secondary host is specified, the connection will be attempted on the primary host only.

This feature is not recommended for implementing redundancy because it only checks for a TCP disconnect. The DataHub Redundancy feature, on the other hand, provides full-time TCP connections to both data sources, for instantaneous switchover when one source fails for any reason. There is no need to start up the OPC DA server and wait for it to configure its data set. You can also specify a preferred source, and automatically switch back to that data source whenever it becomes available. By contrast, the primary and secondary host in the tunnel can act as a primitive form of redundancy, but will only switch on a connection failure at the TCP level, which is only one sort of failure that a real redundancy pair must consider.

- **Local data domain** The data domain in which you plan to receive data.
- **Remote data domain** the master DataHub data domain from which you plan to receive data. Point names will be mapped from the remote data domain (on the master DataHub) into the local data domain (on this DataHub), and vice versa.

Unless you have a good reason for making these different, we recommend using the same data domain name on both DataHubs for the sake of simplicity.

- **Remote user name** The user name for TCP security, established on the tunnelling master, using the DataHub Security option in the Properties window.
- **Remote password** The password for TCP security, established on the tunnelling master, using the DataHub Security option in the Properties window.
- **Secure (SSL)** lets you establish a secure connection using SSL tunnelling as long as the tunnelling master DataHub you are attempting to connect to has been configured for secure connections. The additional options allow for a connection to be made even if the security certificate is invalid, or the host name does not match. We don't recommend using these options unless absolutely necessary. For more about SSL, please refer to the section called “SSL and Firewalls”.

- **WebSocket** lets you connect via WebSocket. This option is applied for both primary and secondary hosts, and allows you to enter a **Proxy address**, a **Proxy port number**, a **username**, and a **password** as needed. When tunnelling through a proxy, HTTP uses normal HTTP proxy, and HTTPS uses HTTP CONNECT proxy. You can select the **Always use HTTP CONNECT** to use it for HTTP as well as HTTPS.

The WebSocket protocol requires a web server to act as an intermediary. So, for this option you will need to use the DataHub Web Server on the tunnelling master DataHub (as explained here).

There is a DataHub running on a Skkynet cloud server that you can connect to for
testing. Here are the parameters you will need to enter for it:

- **Primary Host** demo.skkynet.com
- **Port** Will be set automatically by the system, 80 for WebSocket and 443 for **Secure (SSL)**.
- **Local data domain** cloud
- **Remote data domain** DataPid
- **Remote user name** demo/guest
- **Remote password** guest
- **WebSocket** Must be selected.
- **Secure (SSL)** Optional.

6. You now have several options for the mirrored connection.

![Image of configuration settings]

- **Data Flow Direction** lets you determine which way the data flows. The default is bi-directional data flow between slave and master, but you can effectively set up a read-only or write-only connection by choosing that respective option.

To optimize throughput, check the **Read-only Receive data from the Master, but do not send** option. Only do this if you actually want a read-only connection. If you do not require read-write access, a read-only tunnel will be faster.

- **When the connection is initiated** determines how the values from the points are assigned when the slave first connects to the master. There three possibilities: the slave gets all values from the master, the slave sends all its values to the master, or the master and slave synchronize their data sets, point by point, ac-
c. **When the connection is lost** determines where to display the data quality as "Not Connected"—on the master, on the slave, or neither.

If you have configured **When the connection is initiated** as **Synchronize based on time stamp** (see above), then this option must be set to **Do not modify the data quality here or on the Master** to get correct data synchronization.

d. **Connection Properties** gives you these options

- **Replace incoming timestamp...** lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.

- **Transmit point changes in binary** gives users of x86 CPUs a way to speed up the data transfer rate. Selecting this option can improve maximum throughput by up to 50%.

For more information, please refer to the section called "Binary Mode Tunnel/Mirror (TCP) Connections".

- **Target is an Embedded Toolkit server** allows this slave to connect to an Embedded Toolkit server rather than to another DataHub.

- **Heartbeat** sends a heartbeat message to the master every number of milliseconds specified here, to verify that the connection is up.

- **Timeout** specifies the timeout period for the heartbeat. If the slave DataHub doesn't receive a response from the master within this timeout, it drops the connection. You must set the timeout time at least twice the heartbeat time.

To optimize this setting, please refer to the section called "Tunnel/Mirror (TCP) Heartbeat and Timeout".

- **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

7. Click **OK** to close the **Tunnel/Mirror Master** window. The fields in the **Tunnelling Slave** table of the Properties Window should now be filled in.

8. Click the **Apply** button in the Properties Window. If the master DataHub is running, this DataHub should establish the tunnelling connection, and the **Status** should display **Connected**. You can view the data with the **Data Browser**, or view the connection with the **Connection Viewer**.

Open the Data Browser and select the data domain you requested to mirror. If the master Cogent DataHub has been correctly configured, you should now see all the master DataHub data for that data domain.
Working with Ranges

The Cogent DataHub can send and receive the data contained in an entire range of an Excel spreadsheets. This data is treated as an array, a two-dimensional range of cells as rows and columns. The array can be as big as necessary (within point size limits), or as small as a single cell—at least one row and one column.

Data Format

Excel transmits array data as a tab-and-newline delimited text string of values. Each value in a row is separated by a tab, and each row is separated by a newline character. The string does not contain any information concerning the source range of the array within the spreadsheet.

Getting a Range out of Excel

There are two methods of transmitting a range, or array data, from Excel to the Cogent DataHub. These exactly match the mechanisms used for individual point data: DDEPoke and DDEAdvise.

Using DDEPoke with a Macro

A DDEPoke command can be issued by Excel to send data to the Cogent DataHub based on a trigger within Excel. For this to work, the Cogent DataHub needs to be configured to act as a DDE server and have registered at least one service name. An Excel macro can then issue a DDEPoke to that service, along with a Cogent DataHub data domain name (the DDE topic), a point name (the DDE item) and a value. If the value is of type Range then Excel will automatically format the value as a tab-and-newline separated string.

Example: See the definition of the PutData function in the Excel macro coding examples below.

Using a DDE Advise Loop

When sending data from Excel to the Cogent DataHub using a DDE advise loop, Excel acts as the DDE server and the DataHub acts as the client. To create the advise loop:

1. Open the Cogent DataHub Properties Window (by right-clicking on the DataHub icon in the Windows system tray and selecting Properties).
2. Click the DDE button.
3. Make sure the Act as DDE client box is checked.
4. Click the Add button. This opens the DDE Item Definition window.
5. Type in the following information:
   - **Connection Name** Choose a name to identify this connection. It must be unique among all DDE connections.
   - **Service** Type in Excel (case is not important).
   - **Topic** Type the name of your worksheet file, including the .xls extension, like this: my_filename.xls.
   - **Item Names** These create a mapping between Excel cells and ranges, and Cogent DataHub point names. You may specify a single cell in r1c1 format, a range of cells in r1c1:r2c2 format, a cell name, or a range name as the DDE Item name. For example:
     \[\begin{align*}
     r2c5 & \quad \text{- accesses the cell E2 (second row, fifth column)} \\
     r3c3:r5c9 & \quad \text{- accesses the range C3:I5} \\
     MyRange & \quad \text{- accesses the cell or range that is named MyRange}
     \end{align*}\]

6. Click the **Add** button. The fields **DDE Item**, **Point Name** and **Data Domain** should automatically fill in with some values.
   
   Check the names in the **Point Name** and **Data Domain** columns. If either of them is not what you need, double-click it to select it, and change it.

7. Click **OK** to close the **DDE Item Definition** window. The fields **DDE Connection Name** and **Status** in the Properties Window should now be filled in as well.

8. Click **OK** to close the Properties Window.

9. Enter some values in the range of the spreadsheet you have defined. You should see the array in the Data Browser change accordingly.

**Getting a Range into Excel**

There are two ways to drag and drop data into Excel to create a range, using DDE advise loops. Or you can use DDE Request and macros.

**Drag and drop a group of points into Excel**

Here is how you can collect a group of points in the DataHub and drag them all into Excel, where the data for each point occupies a unique cell.
1. With the Cogent DataHub and DataSim running, open the Data Browser.
2. Select a group of points in the Data Browser.
3. Drag the point names into Excel.

You should see the data updating in the cells.

You can drag and drop point names, timestamps, and other attributes of a point using the Property dropdown list. Please refer to Drag and Drop Style and Property in the Data Browser section for more details.

Drag and drop an array into Excel

Here is how you can take a single point in the DataHub whose value is an array, and have each value in the array occupy a unique cell in Excel.

To demonstrate this, we are going to first combine the two procedures shown above to create an array in the DataHub

Make an array

1. Select a range in Excel, such as created in Drag and drop a group of points into Excel above, and in the name box at the top left corner, enter the name FirstRange.

2. In the Cogent DataHub Properties Window, select the DDE option and make sure the Act as DDE client box is checked. Then click the Add button.
3. In the **DDE Item Definition** window type in the following information:
   - **Connection Name** Type in *Ranges*.
   - **Service** Type in *Excel*.
   - **Topic** Type in *Book1*, or the name of your worksheet file including the *.xls* extension.
   - **Item Names** Type in *FirstRange*.

4. Click the **Add** button. The fields **DDE Item** and **Point Name** should be *FirstRange*, and the **Data Domain** should be *default*.

5. Click **OK** to close the **DDE Item Definition** window, and in the Properties Window click **OK** to close it as well.

6. Open the Data Browser and go to the *default* data domain. You should see the point *FirstRange*, with a value like this:

<table>
<thead>
<tr>
<th>Point Name</th>
<th>Time Stamp</th>
<th>Quality</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstRange</td>
<td>Seq 10 1...</td>
<td>Good</td>
<td>Ramp-0.325Sine0.452413Sine5Square0.36T</td>
</tr>
</tbody>
</table>

The array is now ready to put into Excel.

**Drag and drop the array**

For simplicity's sake we are going to just put the same array back into Excel.

1. Click on the *FirstRange* point name, and drag it into Excel, dropping it in cell D1.

   ![Excel screenshot showing point dragged into cell D1]

2. For older versions of Excel, the values don't start updating right away because you have to tell Excel how to paste in the link.

   ![Excel screenshot showing paste special menu]

   Go to the **Edit** menu and select **Paste Special**.
3. Select **Paste link** and click **OK**. The cells should fill with the correct, updating data.

**Using DDE Request in Excel**

If you are creating macros in Excel to read data from the Cogent DataHub, you can use the `DDERequest` function call. This will return an array type value that can be written directly into any range in the spreadsheet. If the array data is larger in any dimension than the range into which it is written, then extra data in the array is discarded. If the array data is smaller than the target range then extra cells in the range are filled by repeating the data in the array. See below for an Excel macro that dynamically determines the target range to ensure that all array data is entered into the spreadsheet with no duplication.

**Sample Excel Macros for Arrays**

The following macros represent the entire macro set for a simple test spreadsheet that reads and writes a single array point in the Cogent DataHub. The two functions `GetData` and `PutData` can be attached to buttons on a spreadsheet for easy testing. The `PutData` subroutine contains two alternative representations of the source range, one of which is commented out in the macro.

```vba
Sub GetDataArray(Channel As Integer, SheetName As String, DataPoint As String, StartRow As Integer, StartCol As Integer)
    Dim NRows As Integer, NCols As Integer
    ' This sub performs a DDERequest for DataPoint in the DDE Channel
    ' and reads in a tab delimited array with carriage returns at the
    ' end of each line. It then fills a range of cells with the data.
    ' The native format for Excel data is tab delimited text with a
    ' carriage return at the end of each row of data. If we assign
    ' this type of data to a range of cells using the FormulaArray
    ' function, Excel automatically parses the data and fills it into
    ' the specified range. The real trick here is to ensure that the
    ' range is the same size as the incoming data, so we do not have
    ' to know the size a priori.
    ' request DataPoint from Channel
    DataArray = DDERequest(chan, DataPoint)
End Sub
```
' find the upper row and column bounds for the variant array

If StartCol = 0 Then StartCol = 1   ' Starting column where
data will go in our sheet
If StartRow = 0 Then StartRow = 1   ' set the starting row
NCols = 1                           ' set default number or
columns to 1
On Error Resume Next                ' ignore errors (error occurs
if array has one dimension)

' get upper bound of the array columns
' the following line will generate an error if the array is only
' a one dimensional array
' We just skip this, and use the default 1
NCols = UBound(DataArray, 2)

On Error GoTo 0                     ' allow errors
NRows = UBound(DataArray, 1)        ' get upper bound of
' array y dimension

NRows = NRows + StartRow - 1        ' add offset from StartRow
' - this is the ending row
NCols = NCols + StartCol - 1        ' add offset from StartCol
' - this is the ending col

' the following line fills up the cells in the range starting
' in "StartCol:StartRow" to "Nrows:Ncols" with the data from
' the variant array
Sheets(SheetName).Range(Cells(StartRow, StartCol), _
    Cells(NRows, NCols)) = DataArray
End Sub

Sub PutDataArray(Channel As Integer, SheetName As String, DataPoint _,
    As String, StartRow As Integer, StartCol As Integer, _
    NRows As Integer, NCols As Integer)
    DDEPoke Channel, DataPoint, _
    Sheets(SheetName).Range(Cells(StartRow, StartCol), _
        Cells(StartRow + NRows - 1, StartCol + NCols - 1))
End Sub

Sub PutDataRange(Channel As Integer, DataPoint As String, _
    DataRange As Range)
    DDEPoke Channel, DataPoint, DataRange
End Sub

Sub GetData()
This is a test function assigned to a button. It reads a test point into an arbitrarily sized matrix starting at A10

```vba
Dim chan As Integer
chan = DDEInitiate("datahub", "default")
GetDataArray chan, "Sheet1", "TestArray", 10, 1
DDETerminate (chan)
```

```vba
Sub PutData()
' This is a test function assigned to a button. It writes a 3 row x 5 column area of Sheet1 into a single data point in the DataHub. You can use either PutDataArray or PutDataRange, depending on how you wish to specify the range.
' Dim chan As Integer
chan = DDEInitiate("datahub", "default")
"PutDataArray chan, "Sheet1", "TestArray", 1, 1, 3, 5
PutDataRange chan, "TestArray", Sheets("Sheet1").Range("A1:E3")
DDETerminate (chan)
```

**Basic Trouble-Shooting for Excel Connections**

If you cannot get a connection working in Excel, there are a couple of things you can check:

- Is the workbook automatic calculation turned on? It may need to be in order for the links to update.
- Is **Update links to other documents** turned off? It may need to be turned on for the links to update.
- Is **Ignore other applications that use Dynamic Data Exchange (DDE)** turned off?
- In the worksheet, select **DATA—>Edit Links** and ensure that the links are automatically updated. Also select **Startup Prompt** in the **Edit Links** dialog and ensure that the links are set to automatically update.
- In the **Edit Links** dialog, try pressing **Update Values**. Do the values change in the spreadsheet? If they do, it seems to confirm that there is an Excel setting interfering with normal operation.

Also be sure these initial points are covered:

- Make sure the **Act as a DDE Server** checkbox is checked in the DataHub DDE properties window.
• Make sure you have at least one DDE Service name listed in the list just below that checkbox.
• Make sure Excel is set to do **Automatic Calculations**, (See **Tools, Options, Calculation** tab for details). When set to **Manual Calculations**, Excel will not process incoming DDE events.

**Messages from Excel**

When you save and close a spreadsheet connected to the DataHub and then attempt to reopen it, you may get one or more messages, depending on your security settings in Excel, or other circumstances. Here’s a summary of each message, and what to do:

This document contains macros. Enable them?
  
  Click **Enable Macros**.

This workbook contains links. Update them?
  
  Click **Update**. If the DataHub is already running, all the links should then update automatically. If not, you may get a **#REF!** entry in some cells, and the next message (see below) will probably appear.

Remote data not accessible. Start the DataHub?
  
  Click **No**. At this point the best thing to do is close the worksheet, start the requested program, and then reopen the worksheet. When you update the spreadsheet (see above) this time you won’t get any **#REF!** entries.

**Excel not accepting DDE client connection**

When using the DataHub with Excel for the first time, you may encounter this message:

```plaintext
Outgoing DDE Client connection failed...
```

This is likely due to Excel configuration. In Excel, go to the **Options** menu, and select **Advanced**. There, under the **General** heading, ensure that the box **Ask to update automatic links** is not checked.

**Excel not updating**

The default settings in Excel allow you to drag and drop from the Cogent DataHub into your spreadsheet and see the data updating automatically. Sometimes however the Excel configuration may have been changed so that you do not see this. For example, if you
drag a data point into Excel and you get the first value, but then nothing after that, you may want to check the following settings.

1. For newer versions of Excel, in the **File** menu go to **Options > Formulas > Calculation options**.

   a. Ensure that the **Automatic** option in **Workbook Calculation** is selected.
   
   b. Go to **Advanced > When calculating this workbook**

   c. Ensure that the **Update links to other documents** option is selected.

2. In older versions of Excel, from the **Tools** menu, choose **Options** to open the Options window.

   a. Ensure that the **Automatic** option in **Calculation** is selected.
   
   b. Ensure that the **Update remote references** option in **Workbook options** is selected. Then close the **Options** window.

3. From the **Edit** menu, choose **Links** to open the Edit Links window.
4. Ensure that the **Automatic** option for **Update** is selected. Then close the Edit Links window.
DataHub Scripting

The DataHub has a powerful, built-in scripting language called Gamma. Using Gamma, you can write scripts to interact with the DataHub and its data in various ways, such as:

- Attach scripts to specific data points so the scripts are run whenever the point value changes.
- Build custom dashboards and summary displays directly in Gamma scripts to create self-contained DataHub applications.
- Create alarm condition scripts and have them display warning messages to the user.
- Create Excel readable log files from your live data by running logging code on a timed interval, or whenever a point change occurs.
- Connect to ODBC compliant relational databases to extract data as well as create records from live data.
- Apply linear transforms on data as it passes through the DataHub (for example change a temperature reading from Celsius to Fahrenheit).
- Create full simulation programs to test production systems before you 'go live'.

Please refer to the DataHub Scripting manual for more information about scripting.

Tools

The DataHub comes with a built-in Script Editor for writing and editing scripts, as well as a Script Log for viewing script outputs.
Please refer to the DataHub Scripting manual for more information about how to use these tools.

**DataHub ODBC (Open Database Connectivity) Scripting**

The ODBC support in the DataHub provides an interface to any ODBC-compliant database. It lets you create a class for any database table, and assign each column of the table as an instance variable of the class, giving you complete access to any point of data in the database.

Please refer to the DataHub ODBC Support manual for more information.
DataHub Windows Scripting

The DataHub offers Windows scripting support, with the classes necessary to create windows, buttons, frames, tabs, entry fields, and so on—all animated with live data. Here is a screenshot of a test program:

![Screenshot of a test program](image)

Please refer to the DataHub Windows Scripting manual for more information. The code for this example is in the `WindowsExample.g` file included in your distribution.
Working With Data

This chapter gives an overview of how the Cogent DataHub handles data and the various protocols it works with.

Data Points

Each value stored in the Cogent DataHub is called a point. A point has the following attributes:

- **Name** A character string. Currently the only limit on length is internal buffer size, about 1000 bytes by default.
- **Value** An integer, floating-point number, or character string.
- **Time** The date and time of the last significant change to the point's value, confidence, quality or other status information.
- **Quality** The quality of the connection, assigned by the Cogent DataHub for this point, such as Good, Bad, Last known, Local override, etc. Please refer to the quality command documentation for a complete list.
- **Type** A character string representing the Windows variant type of the point value. Please see below for a complete list.
- **Confidence** A value from 0 to 100 that indicates as a percentage the probability that the value shown for the point is actually its true value. This feature can be accessed and changed only by using the API. The Cogent DataHub never uses confidence itself, but carries it for use by client applications.

Creating New Points

The Cogent DataHub automatically creates a point whenever a connecting program tries to read, write, or create a point that doesn't exist. When the point is created, the Cogent DataHub assigns its value, time, quality, and confidence.

It is possible to have the Cogent DataHub create points and assign values to them at start-up. Sometimes referred to as seeding, this is done with supplemental configuration files. Please refer to the section called “Configuration Files” for more details.

Deleting Points

It is not possible to directly delete points from the Cogent DataHub. This is because a connecting process may be using that point. Performance does not suffer if there are unused points in the system, but some users prefer to remove them to just keep things tidy. Should a point no longer be in use or requested by any participating program, when the DataHub is shut down and restarted, the point will no longer appear. For multiple instances of the DataHub connected via a tunnel/mirror connection, all of them must be shut down together to ensure that a point is deleted.
Viewing Data Points

You can view the values of all data points with the Data Browser. You can also create a dump of the DataHub's current point list by issuing this command in the Script Log entry field:

```
datahub_command("(dump "c:/temp/datahub.dump")", 1)
```

Point Size Limits

The Cogent DataHub itself does not limit the size of a point data message. The only limits are those imposed by the operating system, and in Windows there are no such limits. If, however, you intend to share data between with Linux or QNX computers, there is a limit of 64000 bytes for QNX and 128000 bytes for Linux. In any case, bear in mind that very large values will take some time to be transmitted over a network.

Data Types

The DataHub supports the following Windows variant data types:

<table>
<thead>
<tr>
<th>Windows variant type</th>
<th>DataHub string</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VT_BOOL</td>
<td>bool</td>
<td>Boolean (1 = TRUE, 0 = FALSE)</td>
</tr>
<tr>
<td>VT_BSTR</td>
<td>bstr</td>
<td>Text string</td>
</tr>
<tr>
<td>VT_BSTR</td>
<td>string</td>
<td>Text string</td>
</tr>
<tr>
<td>VT_CY</td>
<td>cy</td>
<td>Currency</td>
</tr>
<tr>
<td>VT_DATE</td>
<td>date</td>
<td>Date</td>
</tr>
<tr>
<td>VT_DECIMAL</td>
<td>decimal</td>
<td>Decimal number</td>
</tr>
<tr>
<td>VT_EMPTY</td>
<td>any</td>
<td>No type specified, or unknown</td>
</tr>
<tr>
<td>VT_I1</td>
<td>i1</td>
<td>1-byte signed character</td>
</tr>
<tr>
<td>VT_I2</td>
<td>i2</td>
<td>2-byte signed integer</td>
</tr>
<tr>
<td>VT_I4</td>
<td>i4</td>
<td>4-byte signed integer</td>
</tr>
<tr>
<td>VT_I8</td>
<td>i8</td>
<td>8-byte signed integer</td>
</tr>
<tr>
<td>VT_INT</td>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>VT_R4</td>
<td>r4</td>
<td>4-byte real number</td>
</tr>
<tr>
<td>VT_R8</td>
<td>r8</td>
<td>8-byte real number</td>
</tr>
<tr>
<td>VT_UI1</td>
<td>ui1</td>
<td>1-byte unsigned character</td>
</tr>
<tr>
<td>VT_UI2</td>
<td>ui2</td>
<td>2-byte unsigned integer</td>
</tr>
<tr>
<td>VT_UI4</td>
<td>ui4</td>
<td>4-byte unsigned integer</td>
</tr>
<tr>
<td>VT_UI8</td>
<td>ui8</td>
<td>8-byte unsigned integer</td>
</tr>
<tr>
<td>VT_UINT</td>
<td>uint</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>Windows variant type</td>
<td>DataHub string</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>VT_VARIANT</td>
<td>variant</td>
<td>Variant</td>
</tr>
<tr>
<td>VT_BOOL</td>
<td>VT_ARRAY</td>
<td>bool array</td>
</tr>
<tr>
<td>VT_BSTR</td>
<td>VT_ARRAY</td>
<td>bstr array</td>
</tr>
<tr>
<td>VT_BSTR</td>
<td>VT_ARRAY</td>
<td>string array</td>
</tr>
<tr>
<td>VT_CY</td>
<td>VT_ARRAY</td>
<td>cy array</td>
</tr>
<tr>
<td>VT_DATE</td>
<td>VT_ARRAY</td>
<td>date array</td>
</tr>
<tr>
<td>VT_DECIMAL</td>
<td>VT_ARRAY</td>
<td>decimal array</td>
</tr>
<tr>
<td>VT_I1</td>
<td>VT_ARRAY</td>
<td>i1 array</td>
</tr>
<tr>
<td>VT_I2</td>
<td>VT_ARRAY</td>
<td>i2 array</td>
</tr>
<tr>
<td>VT_I4</td>
<td>VT_ARRAY</td>
<td>i4 array</td>
</tr>
<tr>
<td>VT_I8</td>
<td>VT_ARRAY</td>
<td>i8 array</td>
</tr>
<tr>
<td>VT_INT</td>
<td>VT_ARRAY</td>
<td>int array</td>
</tr>
<tr>
<td>VT_R4</td>
<td>VT_ARRAY</td>
<td>r4 array</td>
</tr>
<tr>
<td>VT_R8</td>
<td>VT_ARRAY</td>
<td>r8 array</td>
</tr>
<tr>
<td>VT_UI1</td>
<td>VT_ARRAY</td>
<td>ui1 array</td>
</tr>
<tr>
<td>VT_UI2</td>
<td>VT_ARRAY</td>
<td>ui2 array</td>
</tr>
<tr>
<td>VT_UI4</td>
<td>VT_ARRAY</td>
<td>ui4 array</td>
</tr>
<tr>
<td>VT_UI8</td>
<td>VT_ARRAY</td>
<td>ui8 array</td>
</tr>
<tr>
<td>VT_UINT</td>
<td>VT_ARRAY</td>
<td>uint array</td>
</tr>
<tr>
<td>VT_VARIANT</td>
<td>VT_ARRAY</td>
<td>variant array</td>
</tr>
</tbody>
</table>

**Data Communication Concepts**

These basic concepts of data communications will help you understand how the Cogent DataHub works.

**Send and Receive Data**

- **Send/write data:** A program sends a value for a data point, and the DataHub records, or writes, the value for that point. This type of communication is synchronous. The send and the write are essentially two parts of a single process, so we use the terms pretty much interchangeably. You can write a value to the DataHub manually using the Data Browser.

  A typical write command from a program using DDE protocol is `DDEPoke`.

- **Receive/read data:** A program requests to receive the value of a data point. The DataHub then responds by sending the current value of the point. We call this reading the value from the Cogent DataHub. Again, we sometimes use the two terms interchangeably, and again, this type of communication is synchronous.
A typical read command from a program using DDE protocol is **DDERequest**.

- **'Automatic' Receive**: It is possible to set up live data channels, where a program receives updates on data points sent from the Cogent DataHub. How it works is the program sends an initial request to the DataHub to register for all changes to a data point. The DataHub immediately sends the current value of the point, and then again whenever it changes. The DataHub can receive data automatically in a similar way. This asynchronous type of communication is sometimes referred to as *publish-subscribe*.

  A **DDEAdvise** command sets up this type of connection, which is called an *advise loop*.

**Client - Server**

Exchanging data with the Cogent DataHub is done through a client-server mechanism, where the *client* requests a service, and the *server* provides the service. Depending on the programs it interacts with, the DataHub is capable of acting as a client, as a server, or as both simultaneously.

The client-server relationship itself does not determine the direction of data flow. For example, a client may read data from the server, or it might write data to the server. The data can flow either way; the client might initiate a read or a write, and the server would respond.

**Synchronous and Asynchronous Communication**

Every type of communication, natural or man made, comes in two basic forms: *synchronous* or *asynchronous*.

- Synchronous communication means that for each message, the sender expects to get a reply from the receiver, like a telephone call. There is a back-and-forth exchange, so that each party knows that the other is receiving the message. If there is no response, you can be pretty sure that communication didn't occur.

- Asynchronous communication means that a message gets sent but the receiver is not expected to reply, like a radio broadcast or a newspaper.

  Each of these communication types has its own value and purpose in data communications, and the Cogent DataHub is capable of both. The specific circumstances and application will determine which form of communication you end up using.

**Data Exchange Protocols**

The Cogent DataHub relays data between programs using **OPC**, **TCP** or **DDE**. It also tunnels data over a network or the Internet using TCP. This section gives an overview of these protocols.

**OPC Protocol**

OPC is an interface specification for data communications that is popular in industrial environments. Please refer to the [general information about OPC](https://skkynet.com).
OPC connections are always client-server. Setting up the DataHub to use OPC is simply a matter of configuring it to act as a client or as a server, or both. When acting as a client, it will automatically attempt to find or start the OPC server that has been configured, and then start receiving data. When acting as a server, it will automatically respond to requests from any OPC client on the system.

OPC Items and Properties

There are two implementation of OPC for real-time data access: OPC DA (Classic) and OPC UA. Both use the concept of an item as a way to structure data. In OPC DA, every item has 6 required properties: Value, Timestamp, Quality, Access Rights, Scan Rate, and Canonical Type. The OPC UA spec includes this kind of item, as well as many others. The Cogent DataHub implements only the part of the OPC UA spec that is similar to OPC DA. Furthermore, as users of the data, we are mostly interested in Value, Timestamp, and Quality. OPC items can also have up to 30 optional properties, such as Description, Engineering Units, High, Low, Alarm Level, and so on. For example, an item might represent a temperature reading on a tank like this:

<table>
<thead>
<tr>
<th>Property</th>
<th>Current value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>47.2</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Apr 27 16:27:24.300</td>
</tr>
<tr>
<td>Quality</td>
<td>Good</td>
</tr>
<tr>
<td>High</td>
<td>60.0</td>
</tr>
<tr>
<td>Low</td>
<td>38.5</td>
</tr>
<tr>
<td>Alarm Level</td>
<td>54.0</td>
</tr>
<tr>
<td>Engineering Units</td>
<td>Celsius</td>
</tr>
<tr>
<td>Description</td>
<td>Temperature of Tank A</td>
</tr>
</tbody>
</table>

The Cogent DataHub maintains an item and all of its optional properties as separate data points. The item’s 6 required properties are maintained internally, but the DataHub displays the information corresponding to the Value, Timestamp, and Quality in the Data Browser under the columns Value, Date and Quality.

This relatively simple picture becomes more complex when we learn that the OPC specification allows a property to be an item in its own right. This implies, in turn, that properties can have properties. Some OPC servers implement properties as items, and some do not. Normally when an OPC server does treat properties as items, those items have only the 6 required properties and you don’t get an infinite recursion.

This has implications for configuring the DataHub and working with data sets. If an OPC server implements properties as items, the DataHub could potentially have access to many more data points than for an OPC server whose items are not properties. For this reason, the DataHub makes it optional to pick up all items that are properties.

It also affects the results of a filtered connection to an OPC server. Filters set up in the DataHub are based on items. If your OPC server implements properties as items, and if
you choose to pick up all items that are properties, then any filter you run apply to those
items as well.

**DDE Protocol**

DDE (Dynamic Data Exchange) is a well-established mechanism for exchanging data
among processes in MS-Windows. There are three DDE commands for establishing com-
munication with Windows programs such as Excel. Which of these commands you use de-
pends on how you plan to control the flow of data between the spreadsheet and the Co-
gent DataHub.

1. **DDEPoke** writes a data value to the Cogent DataHub. For example, to send a value
   from an Excel spreadsheet to the DataHub, the **DDEPoke** command is run from within
   an Excel macro. For more details, please refer to the section called “Method 2 - Writing
   Excel macros that use the **DDEPoke** command”.

2. **DDERequest** reads a data value from the Cogent DataHub. To get that value into
   Excel, for example, the **DDERequest** command is run from within an Excel macro.
   For more details, please refer to the section called “Method 2 - Excel Macros using
   **DDERequest**”.

3. **DDEAdvise** creates a connection, called an *advise loop*, that updates a new data value
   automatically. The advise loop is a unidirectional link, established by a client program
   that wants to receive data from a server program. The client continues to receive new
   point values as long as the two programs are running, or until the advise loop is termi-
nated.

You can use **DDEAdvise** to read data from the Cogent DataHub by configuring the
DataHub to act as a **DDE Server**. For an example using Excel, please refer to the sec-
tion called “Method 1 - Drag and Drop using **DDEAdvise**”.

Likewise, you can use **DDEAdvise** to write data to the Cogent DataHub, by configuring
it to act as a **DDE Client**. For an example using Excel, please refer to the section called
“Method 1 - Configuring **DDEAdvise** loops in the Cogent DataHub”.

For more information on DDE, please refer to Appendix G, **DDE Overview**.

**Tunnelling/Mirroring**

The Cogent DataHub uses **DHTP** to communicate over a LAN, WAN, or the Internet. You
can join two or more copies of the Cogent DataHub together and share exact copies of the
data through data **tunnelling/mirroring**. Tunnelling/Mirroring means that the data and any
updates to that data on one DataHub are exactly tunneled/mirrored across the network
onto any other DataHub that is connected. Once a tunnelling/mirroring connection is es-
tablished each participant maintains and updates an identical data set, as simultaneous-
ly as the TCP connection will permit. For more information on tunnelling/mirroring, please
refer to the section called “Tunnel/Mirror”.
DHTP - The DataHub Transfer Protocol

The DataHub Transfer Protocol (DHTP) is used by the DataHub Tunnel/Mirror feature, as well as SkkyHub, ETK, and connected clients to send and receive data in real time over TCP across a LAN, WAN, or the Internet. Originally built upon HTTP, DHTP also supports SSL and WebSocket protocols. In continuous development for over 20 years, DHTP is open and documented in two parts, as the DataHub APIs and the DataHub Command Set.

Each DataHub connected by DHTP requires its own license. License verification is done between DataHubs over the network. Occasionally a slow network may result in misleading "no license" errors. Please refer to TCPLicenseTimeoutSecs for more information.

Additionally, the Cogent DataHub supports various protocols that are native to commonly used industrial applications, like ODBC, OPC, Modbus, etc. The ETK supports OPC UA and Modbus.

Examples

As shown in the above diagram, DHTP may be used for the following connection types:

• DataHub to DataHub for DMZs and tunnelling on LANs and WANs
• DataHub to SkkyHub for OT to IT connections and access to remote locations
• ETK to DataHub for on-premise connections and edge processing
• ETK to SkkyHub for direct connections to the cloud and web HMI
• Custom programs to DataHub, to integrate virtually any application

Applied DHTP Features

SkkyHub and DataHub use DHTP to provide these important Industrial IoT features:
• **Low Bandwidth & Low Latency:** Consumes minimal bandwidth, while functioning with the lowest possible latency

• **Ability to Scale:** Can support hundreds or thousands of interconnected data sources and users

• **Real-Time:** Adds virtually no latency to the data transmission

• **Intelligent Overload Handling:** A broker (DataHub, SkkyHub or ETK) responds appropriately when a data user is unable to keep up with the incoming data rate

• **Quality of Service:** Guarantees consistency of data, preserved through multiple hops

**DHTP Protocol Features**

DHTP communications between and among Cogent DataHub, SkkyHub, ETK, and their clients meet the following criteria for secure, robust industrial and IIoT data communications:

• **Closed Firewalls:** Keeps all firewall ports closed for both data sources and data users

• **Interoperable Data Format:** Encodes the data so that clients and servers do not need to know each others’ protocols

• **Can Daisy Chain Servers:** Multiple instances of brokers (DataHub, SkkyHub or ETK) can be connected to support a wide range of collection and distribution architectures

• **Propagation of Failure Notifications:** Each client application can know with certainty if and when a connection anywhere along the data path has been lost, and when it recovers

• **Simple:** Message syntax is simple enough to be implemented even on resource-constrained devices

• **Streamable:** Messages can be concatenated and streamed without requiring intervening acknowledgements. This allows clients and servers to communicate asynchronously, reducing latency and significantly improving throughput

**The DataHub API**

The DataHub APIs for C++, Java, and .NET lets any TCP-enabled program interface with the DataHub.

**Data Organization**

The Cogent DataHub offers a hierarchical system to organize your data. The hierarchy is separate from the actual data. It contains points, but is not made up of data, per se. You specify the data model, similar to specifying a class in a programming language, and then you create zero or more instances of that data model. The data hierarchy can be viewed in the left-hand pane of the Data Browser window.
The highest level of the data hierarchy is the data domain. You can create as many data domains as you need, and use them to separate data by user, function, or any other criteria. Points in different data domains can have the same name because each data domain creates a separate namespace. Two data domains you are probably familiar with are default, the default data domain, and DataSim which holds data from DataSim. All the data domains in the DataHub are listed in the General option of the Properties window.

The written syntax used by the DataHub to denote data domains and points is:

domain:point

In many cases, this is the only level of data organization you will ever need. However, should you desire a more sophisticated way to structure your data, the DataHub provides a way.

**Assemblies, Subassemblies, Attributes, and Properties**

Within a data domain, data can be arranged hierarchically as assemblies, subassemblies, attributes, and properties. Each assembly can have zero or more attributes and zero or more subassemblies, and each attribute can have zero or more properties. Subassemblies can have subassemblies. You can think of assemblies and subassemblies as branches in a tree, and attributes as the leaves. Here is an example of what a tree might look like:

```
Data Domain
  Assembly
    Subassembly (zero or more)
      Attribute (zero or more)
        Property (zero or more)
    Attribute...
    Attribute...
```
Attributes describe the attributes in more detail. An attribute can have a default property such that if you interact with the attribute point directly you will in fact be interacting with its default property. For example, an item might be plant.temperature, with properties value, highlimit, units. This would create 4 tags:

```
plant.temperature
plant.temperature.highlimit
plant.temperature.units
plant.temperature.value
```

The tags:

```
plant.temperature
plant.temperature.value
```

are aliases of one another. Both refer to the default property of plant.temperature. If you specify no property at all for an item, the item takes on the default property.

### Attributes and Types

It is common for attributes to contain the same type of information. For example, all temperatures in a system are likely to share units, high alarm level, and value. To avoid repeating this information for each and every temperature in the system, we use a type. A type is the prototype, or class, of an attribute. You define a type and its properties first, and then define attributes of that type on assemblies. When the assembly is instantiated, its attributes are instantiated by creating an attribute and then assigning the properties to it that are associated with the attribute's type.
There is an alternative to using types and attributes as described here, a *private attribute*. A private attribute provides a one-command (*private_attribute*) means of creating an attribute on an assembly without having to define a type. This prevents the attribute properties from being shared across more than one attribute in the assembly or in other assemblies, but is easier to use when defining simple hierarchies. (See Example 2.)

**Example 1: Attributes and Types**

Suppose we want to create a hierarchical data model like this: We have a control system consisting of process areas, that we will call "Plants". Each plant contains 2 boilers, and each boiler has a pump and 2 valves. Each boiler measures temperature, pressure and level. Each pump measures speed, on/off state and operating temperature. There are two types of valves - a normal one that only measures position, and another that also verifies that it has power applied to it. Temperatures have a value and a high alarm limit.

The sample file `plant.cfg` shown below included in the DataHub distribution will create a point hierarchy in the default data domain that looks like this:

![Hierarchical data model](image)

You can have the DataHub load this `plant.cfg` configuration file on startup (see the section called “Configuration Files”).

```plaintext
;;; Create a generic object to share all of the common properties and attributes in the model. Give it a common property, called "name"
(assembly default Object)
(property default Object AUTO name string rw "unnamed" 100)

;;; Create a temperature attribute to be shared by boilers and pumps. It has three properties: value,highlimit,units
(type default Temperature)
(property default Temperature AUTO value R8 rw 0 100)
(property default Temperature AUTO highlimit R8 rw 120 100)
(property default Temperature AUTO units STRING rw "C" 100)
(defaultprop default Temperature value)
```
;;; Create a pressure attribute for boilers.

(type default Pressure)
(property default Pressure AUTO value R8 rw 0 100)
(property default Pressure AUTO units STRING rw "kPa" 100)
(defaultprop default Pressure AUTO value)

;;; Create a plant model, sharing the properties and attributes
;;; of "object"

(assembly default Plant Object)

;;; Create a boiler model, as an "object"

(assembly default Boiler Object)
(attribute default Boiler temperature Temperature)
(attribute default Boiler pressure Pressure)

;;; Create a pump model, as an "object"

(assembly default Pump Object)
(attribute default Pump temperature Temperature)
(property default Pump AUTO speed R8 rw 0 100)
(property default Pump AUTO state I4 rw 0 100)

;;; Create a valve object. It has a property, position, directly
;;; attached to the assembly. We do not need an attribute unless
;;; there is more than one property to be associated with it. In
;;; this example position has only a value, without limits or
;;; units.

(assembly default Valve Object)
(property default Valve AUTO position R8 rw 0 100)

;;; Create a specialization of a Valve that also measures whether
;;; the valve is powered.

(assembly default Powervalve Valve)
(property default Powervalve AUTO powered I4 rw 0 100)

;;; Create the hierarchy in the model

;;; Plants have two boilers, named boiler1 and boiler2
(subassembly default Plant Boiler boiler1)
(subassembly default Plant Boiler boiler2)
Boilers have one Pump, named pump
(subassembly default Boiler Pump pump)

Boilers have a normal valve and a powered valve, named valve1
and valve2 respectively.
(subassembly default Boiler Valve valve1)
(subassembly default Boiler Powervalve valve2)

Create two plants named plant1 and plant2. These actually
create the data points in the DataHub and arrange them in
the hierarchy specified above. Up to this point, the commands
have just been building a model. These calls instantiate the
model.
(instance default plant1 Plant)
(instance default plant2 Plant)

Example 2: Private Attributes

Here is a simpler example for creating a hierarchical data model using private attributes. This models a control system in a factory with 2 pumps. The sample file shown below will create a point hierarchy in the default data domain that looks like this:

Create two assemblies.
(assembly testdomain factory)
(assembly testdomain pump)

Create two subassemblies.
(subassembly testdomain factory pump pump1)
(subassembly testdomain factory pump pump2)

Assign private attributes.
(private_attribute testdomain pump flow R8 rw 0 100)
/private_attribute testdomain pump amps R8 rw 0 100)
/private_attribute testdomain pump onoff BOOL rw 0 100)
; Instantiate the model.
(instance testdomain factory1 factory)
Optimizing Data Throughput

The Cogent DataHub has a wide range of configuration options. Among these there are several settings that will optimize data throughput. These are explained in this chapter.

Binary Mode Tunnel/Mirror (TCP) Connections

TCP/IP connections to the Cogent DataHub can be either ASCII or binary mode. A large part of the CPU cost of transmission is marshalling messages (constructing messages at the source and parsing them at the destination). The binary mode is more efficient in both network bandwidth and CPU usage for both the sender and the receiver. Binary mode requires that the CPU architecture of the sender and the receiver agree, so you can only use this mode if you are running both the sender and the receiver on an Intel x86 CPU. The CPU gain could be as much as 50% when using binary mode. Numeric data benefits most from this option.

How to Optimize

• For tunnelling connections, always use binary mode if possible. Please refer to How to Optimize, Binary mode transmission for details.
• For TCP/IP connections using the C++ API, always use binary mode if possible. Please refer to How to Optimize, the section called “DataHub C++ API” for details.
• Always use binary mode when writing programs in Linux or QNX with the Cogent C API.

Tunnel/Mirror (TCP) Heartbeat and Timeout

The Cogent DataHub uses a heartbeat to determine the status of the network connection. The tunnel/mirror slave sends a special heartbeat message to the master at specified time intervals, to detect network failures. If the master does not respond within a certain timeout period, the slave changes the status of its connection to Disconnected and attempts to reconnect.

The overall principle for optimizing the heartbeat and timeout is choosing settings based on the dynamics of the system and how quickly you need to know about a lost connection. With that in mind, we recommend choosing the longest reasonable timeout that fits your needs, and then set the heartbeat to half of that.

Specific Guidelines

1. The timeout should be at least twice the heartbeat.
2. The default heartbeat is set to 1 second and the timeout to 5 seconds. That is reasonable LAN timing. If the network or CPU situation could produce temporary timeouts longer than 5 seconds then this timing could cause unnecessary disconnect/reconnect cycles.
3. When connecting to SkkyHub over the Internet we recommend setting the heartbeat to 10 seconds and the timeout to 30 seconds. That generally produces a very stable connection.
4. When connecting to a remote system via a metered connection (usually cellular) we recommend setting the heartbeat to 30 seconds and the timeout to 60 seconds.

5. A short timeout will notify the DataHub quickly when a connection is lost. If you need to know quickly about a lost connection, set the timeout low.

6. A long timeout makes it less likely that the connection will time out during a temporary event, like a burst of heavy network traffic or a high CPU load on one of the computers. A short timeout under these conditions will cause extra traffic to reconnect the tunnel, and will cause data to temporarily become Not Connected.

7. A short heartbeat will cause more traffic when the connection is idle. This traffic is not significant on a LAN, but could be important on a metered connection like a cellular modem. If traffic is an issue, set the heartbeat high.

**Additional Considerations**

Normal ping times on a LAN are less than a millisecond, so you could technically set the heartbeat very low on a LAN. However, Windows O/S time slices are 33 ms, which means that a busy CPU could easily hold off the DataHub for tens of milliseconds. We generally do not recommend sub-second heartbeats without a really good reason.

VPNs, proxies and encryption can all have an impact on latency, which can result in transmission delays.

**Old Value Queuing**

The Cogent DataHub maintains a queue of old values for all registered points for each client. The depth of this queue is variable. The purpose of queuing old values is to reduce the chance that a data change will be missed during bursts of abnormally high data flow.

For example, if a switch is turned on, then off, then on again very rapidly, the data might arrive at the DataHub so quickly that it has no opportunity to send it where it needs to go before the next value arrives. If this happens, the DataHub may only transmit the final "on" value and the client will not notice that there was in fact an on-off-on transition.

The Cogent DataHub can maintain a short queue to reduce the probability of this happening. If the queue is at least three values deep, the DataHub will send the on-off-on transition even if it knows that the first two values are already stale.

Old value queuing is harmless so long as periods of abnormally high data flow are short. If the data flow rate is high enough that the DataHub can never keep up, the effect is that the old value queue will always be full, no matter how long or short it is. The CPU cost of maintaining even a short queue in a sustained overload situation is very high, and depends on the queue depth. See also the section called “CPU Saturation”.

**How to Optimize**

- If your system runs at CPU saturation, eliminate the old value queue if at all possible for TCP/IP connections. Please refer to How to Optimize, Old value queueing and un-buffered delivery for details.
• If your system runs at CPU saturation, eliminate the old value queue if at all possible for the Gamma scripting engine. Please refer to How to Optimize, the section called “Gamma scripts” for details.

Un-Buffered Delivery

The Cogent DataHub buffers data that will be transmitted to a client such that if it knows that more incoming data is available, it will hold off outgoing transmissions until it has a complete data set to send onward to the client. This does not introduce extra latency because the DataHub will only accumulate data destined for a client that arrives together in an incoming message.

This buffering greatly increases efficiency by reducing thread context switching and by giving its protocol-specific data transmitters an opportunity to collect more than one data change into a single outgoing message.

One of the side-effects of buffering is that the old value queue will be more likely to fill. If your application is very sensitive to every change of value, then it may be necessary to turn off the buffering. The CPU penalty for turning off buffering is very high, perhaps as much as 200% in heavy load conditions.

How to Optimize

Do not use un-buffered data delivery in high load conditions unless you absolutely must. Please refer to How to Optimize, Old value queueing and un-buffered delivery for details.

Screen Output

Output to the screen can use a huge amount of processing time. The worst offenders are multi-line text boxes where text is constantly being added and scrolled. If you think that the CPU usage for the Cogent DataHub is too high, close all Event Log and Script Log windows.

You may also find that the Data Viewer window uses too much CPU if the data is changing very rapidly or if the number of data points visible in the right-hand pane of the Data Viewer is very large. Try closing the Data Viewer to reduce CPU load. If you need to monitor some data in the Data Viewer, consider using bridging to collect the subset of data points that you need into separate data domain. Viewing that subset of the data will consume less CPU and may have the added benefit of being more convenient.

How to Optimize

• Close the Event Log windows when not in use.
• Close the Script Log window when not in use.
• Close the Data Viewer when not in use, or use bridging to create a smaller subset of data in a separate data domain.

CPU Saturation
When your system is running with maximum CPU utilization, the transmitting and receiving threads within the Cogent DataHub must all share the CPU available. This will cause data to be queued more frequently, and will cause the DataHub to take measures to cope with the lack of CPU. The DataHub treats a high-CPU condition as if the various connections cannot consume data as quickly as it is available. It will begin tracking and ultimately discarding old data values, and will more aggressively accumulate data changes into larger messages wherever possible. The DataHub’s goal is to ensure that latency remains low, that all clients continue to receive data, and that the clients always receive the most recent data that is available.

This effort to cope with reduced resource availability also uses more CPU, somewhat further increasing the system load. If old value queues are deep, the system can reach a “tipping point” from which it is difficult to recover without severely reducing the input data rate.

**How to Optimize**

Avoid running your system at maximum CPU capacity.

**How to Optimize**

**Tunnel/Mirror (TCP) connections**

**Read-only connections**

On the slave side of the connection, you can determine the data flow direction for the connection. If the data flow is to be one-way, from the master to the slave (i.e. the slave will only read from the master, not write), you will get the fastest performance by configuring the connection “read-only”, as follows:

1. On the DataHub that is making the slave side connection, right click on the DataHub system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror.
3. In the Data Flow Direction section, select Read-only: Receive data from the Master, but do not send.
4. Click Apply.

**Binary mode transmission**

Ensure that the slave side of the tunnel/mirror connection is set to use binary mode transmission:
1. On the DataHub that is making the *slave* side connection, right click on the DataHub system-tray icon and choose **Properties**.

2. In the Properties window, select **Tunnel/Mirror**.

3. In the **Tunnel/Mirror Slave** section, highlight the host name of the tunnelling master, and click the **Edit** button to open the Tunnel/Mirror Master Configuration window.

4. Check the **Transmit point changes in binary** box.

5. Click **Apply**.

For more information, please refer to the section called “Binary Mode Tunnel/Mirror (TCP) Connections”.

**Timeout and Heartbeat**

To optimize performance for the network heartbeat and timeout, please refer to the section called “Tunnel/Mirror (TCP) Heartbeat and Timeout”.

**Old value queueing and un-buffered delivery**

Configure the **master** side of the tunnel/mirror connection:

1. On the DataHub that is the *master* for the tunnel/mirror connection, right click on the DataHub system-tray icon and choose **Properties**.

2. In the Properties window, select **Tunnel/Mirror**.

3. In the **Tunnel/Mirror Master** section, you can choose between one of three states:
   - **No queuing with buffered delivery** This is the fastest state. To do this, un-check the option **Try to send data even if it is known to be superseded** in the Tunnel/Mirror configuration tab:

   ![No queuing with buffered delivery configuration](image)

   - **Queuing with buffered delivery** This is a reasonable compromise that will keep up to three old values for each data point if there is a short burst of heavy data traffic. If there are more than three values for a point outstanding, the oldest will be discarded. To do this, check the option **Try to send data even if it is known to be superseded**, but do not select any of the data types below it:

   ![Queuing with buffered delivery configuration](image)
• **Queuing with un-buffered delivery** This is the slowest mode, but also the one least likely to discard old values during short periods of heavy data traffic. In this mode there is a queue of up to 3 old values for each data point. In addition, you may choose to force the data transmission when a point with any of the specified types changes. The goal here is to possibly allow buffering (and hence, possible discarding of old values) for some types, while attempting to preserve all changes for other types. For example, here we have chosen to force an outbound transmission if any boolean or string value changes, but to buffer any changes in floating point and integer types:

![Checkbox interface for selecting types](image)

4. Click **Apply**.

For more information, please refer to the section called “Old Value Queuing” and the section called “Un-Buffered Delivery”.

**DataHub C++ API**

**Binary mode connections**

- In your program, call the method

  ```cpp
  CDataHubConnector::sendBinaryPointMessage(bool enable)
  ```

  to enable or disable binary messages.

For more information, please refer to the section called “Binary Mode Tunnel/Mirror (TCP) Connections”.

**Gamma scripts**

The Gamma engine services all DataHub scripts through a single queue, for the sake of efficiency. This means that any changes you make to the default behaviour will apply to all running scripts. By default, the Gamma engine runs with a 3-deep queue and buffered transmission. This is equivalent to the queuing with buffered delivery option (above) for TCP/IP connections. As of Cogent DataHub version 6.4.2, you can modify the behaviour of the queuing and the buffering via Gamma function calls:

- **set_point_queue_depth** lets you specify the depth of the per-point queue. It is wise to keep this value small. Please see `set_point_queue_depth` in the DataHub Scripting manual for details about this function.

- **get_point_queue_depth** determines the current point queue depth for Gamma. Please see `get_point_queue_depth` in the DataHub Scripting manual for details about this function.

- **set_point_flush_flags** sets which data types will cause the point buffer to immediately be transmitted to the Gamma engine. Please see `set_point_flush_flags` in the DataHub Scripting manual for details about this function.
Using DataHub Commands

The DataHub has an internal command set, documented in the Cogent DataHub Command Set reference. When you change the configuration of the DataHub in the Properties window, one or more of these commands is written in the configuration file, and the DataHub receives that command every time it starts up. You can use these commands to create custom configuration files. (Please see the section called “Configuration Files”.)

It is also possible to issue these commands to the DataHub during run-time in any of the following ways:

- With a DataHub script, using the special `datahub_command` function.
- Using the DataHub APIs for C++, Java, and .NET.
- Over a direct TCP connection.

This is how custom applications can interact directly with the DataHub. For example, DataSim connects to the DataHub by using the DataHub APIs for C++, Java, and .NET.

Command Syntax

DataHub commands have the following syntax:

```
(command arg1 arg2 ...)
```

The whole command must be surrounded by parentheses. The command name and its arguments are each separated by white space—single spaces, tabs, or carriage returns are allowed. For example, the following line of a custom configuration file tells the Cogent DataHub to create a new data domain, named TestDomain.

```
(create_domain TestDomain)
```

Multiple-word strings must be in quotes. Numbers take their own values. Booleans are 0 for false and 1 for true.

Return Syntax

When the Cogent DataHub executes a command, it may return a success message or an error message. The returned message contains the original command, and some or all of the arguments for the command. These messages will be received by programs using the DataHub APIs for C++, Java, and .NET, but they will not be received by DataHub scripts. The two types of success messages are:

- **No arguments** (success command)
- **One or more arguments** (success command arg1)

Success messages are returned if the DataHub command `acksuccess` has been previously issued with a value of 1. A value of 0 means no success messages will be returned. Error messages are returned any time there is an error. There are four types of error messages:
• **No arguments** (error "-2: (command): error message")

• **One argument** (error "-2: (command arg1): error message")

• **Two arguments** (error "-2: (command arg1 arg2): error message")

• **More than two arguments** (error "-2: (command arg1 ...): error message")

The error messages are the negation of these error codes:

- **ST_OK** The function executed without error.
- **ST_ERROR** An error occurred.
- **ST_NO_TASK** A required task does not exist.
- **ST_NO_MSG** There is no message available.
- **ST_WOULD_BLOCK** This action would block, and is not permitted.
- **ST_INTR** An interrupt occurred.
- **ST_FULL** The queue is full.
- **ST_LOCKED** A DataHub point is locked.
- **ST_SECURITY** The security level is insufficient.
- **ST_NO_POINT** A required DataHub point does not exist.
- **ST_INSIG** A change in a DataHub point's value is insignificant. This is not really an error, but a notification that no exception will be generated by the DataHub.
- **ST_UNKNOWN** There is an unknown error.
- **ST_NO_QUEUE** A target task has no queue, or `qserve` is absent.
- **ST_CMD_SYNTAX_ERROR** The command was not found, or there was a syntax error.
- **ST_REPLIED** The reply was complete.
- **ST_WRONG_TYPE** The type of a point or variable was wrong.
- **ST_TOO_LARGE** A value to be written to memory is larger than the available buffer.
- **ST_NO_MEMORY** There is insufficient memory available.
- **ST_OLD_DATA** Time-significant data is out of date.
- **ST_TIMEOUT** A timeout occurred in poll mode.

**Sending Commands by TCP**

It is possible to send commands directly to the DataHub over TCP, by using a tunneling/mirroring connection. The communication between a client and the DataHub over TCP follows the guidelines below.

• The connection is a single bi-directional socket.
• All communication from the DataHub to the client is non-blocking and asynchronous. It is possible to use blocking I/O in the client, and to wait for a response from the DataHub as well, but we don’t advise doing either.

• Since TCP is streamed, there are no packets as such. Each message starts with an open parenthesis and ends with a matching closing parenthesis. Messages should be terminated with a newline (\n) character. If they are not, then the DataHub will hold off on processing the incoming messages until it receives a newline character, and then it processes all messages in order, in a batch. There is a maximum character length allowed (around 1 MB) for a batch of messages, after which the DataHub will discard data until it sees a newline.

• Parameters within a message can themselves be parenthesized expressions. For example, the following message contains a command and five parameters:

```
(OPCAddItem server1 item1 0 default:server1.item1
 (default server1 item1))
```

The fifth parameter is itself a command: (default server1 item1).

• All strings in a message are surrounded by double quotes. Inside the double quotes, the sequence " embeds a double quote, \ embeds a \ character, \n embeds a newline, \t embeds a tab, \f embeds a form feed, \r embeds a carriage return. \ followed by any other character produces that character with the \ removed. Parentheses inside double quotes do not match parentheses outside double quotes.

• You can embed any character inside a non-quoted string by putting a \ in front of that character, so the string abc\ def would be the same as "abc def".

• Example of using a TCP socket directly without going through the Cogent C API:

```
SOCKET s;
char buf[256];
int len;
char *pointname = "testpoint";
double value = 1.0;

len = sprintf(buf, "(cset "default:%s" %g)\n", pointname, value);
send(s, buf, len, 0);
```

There is a function in the C++ header file that parses this kind of stream, called UT_LispParse, and another function called UT_LispString that can help a little with writing Lisp expressions. They automatically add double-quotes around %s formatted strings, and escape characters within the string.
Cogent DataHub Command Set

This is the internal command set for the DataHub. For general information on how to use these commands, please refer to *Using DataHub Commands.*
acksuccess

acksuccess — tells the DataHub to return success messages.

Synopsis

(acksuccess 0|1)

Arguments

0 | 1

Use 1 to have messages returned, or 0 to not have messages returned.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command tells the DataHub to return a message for successfully executed command. Error messages are always sent for unsuccessful attempts to execute a command. Please refer to Return Syntax for details about the message format.
add

add — adds a value to a point.

Synopsis

```
(add name number [secs] [nano])
```

Arguments

- **name**
  The name of a point, which must be a number type.

- **number**
  A value to add to the value of the point.

- **secs**
  The time in seconds. If this is not specified, the current date and time in seconds is used.

- **nano**
  A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to add to the value of a point.
**alias**

*alias* — creates an alias point for an existing point.

**Synopsis**

```
(alias point alias)
```

**Arguments**

- `point`
  The name of a DataHub point, as a string.
- `alias`
  A name for the new point, as a string.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command creates a new point that will alias, or tunnel/mirror its value with, an existing point. The first point must exist, and the second point will be created if it does not exist. They do not need to be in separate domains.
alive

alive — tells the DataHub that the client is running.

Synopsis

(alive)

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command provides a means for the client to tell the Cogent DataHub that it is still up and running.
append

append — appends a string to the value of a point.

Synopsis

(append name string [secs] [nano])

Arguments

name
The name of a point, which must be a string type.

string
A string to add to the current string value of the point.

secs
The time in seconds. If this is not specified, the current date and time in seconds is used.

nano
A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to add a string to the value of a point.
assembly

assembly — creates an assembly.

Synopsis

(assembly domain name [supername])

Arguments

domain
   The name of the domain in which this assembly will be created.

name
   A name for this assembly.

supername
   The name of an assembly.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates an assembly level of a data organization. The supername denotes a special assembly that can be created to hold properties and attributes that may be used by several assemblies, in much the way that a parent class has instance variables that are used by child classes. For more information about assemblies and an example, please refer to the section called “Assemblies, Subassemblies, Attributes, and Properties”.
**attribute**

*attribute* — creates an attribute.

**Synopsis**

```
(attribute domain assemblyname attrname typename)
```

**Arguments**

- **domain**
  The domain in which this attribute applies.
- **assemblyname**
  The assembly or subassembly in which this attribute applies.
- **attrname**
  The name of this attribute.
- **typename**
  The type of this attribute.

**Returns**

A message indicating success or error. Please refer to [Return Syntax](#) for details.

**Description**

This command creates an attribute. For more information and an example, please refer to *Assemblies, Subassemblies, Attributes, and Properties*.
auth

auth — requests authentication for a client.

Synopsis

(auth username password)

Arguments

username

A user name in plain text. Any non-alphanumeric characters must be in double quotes.

password

A password in plain text. Any non-alphanumeric characters must be in double quotes.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is sent to the DataHub by a client to request authentication.
authreload

authreload — is new, not yet documented.

Synopsis

(authreload)

Description

This command has not yet been documented.
auto_create_domains

auto_create_domains — automatically adds domains requested by clients.

Synopsis

```plaintext
(auto_create_domains 0|1)
```

Arguments

0 | 1

Use 1 to have domains added, or 0 to not have domains added automatically.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command instructs the DataHub to create a domain automatically if a client requests a domain that doesn't already exist. This corresponds to the Automatically add domains requested by clients checkbox in the General option of the Properties window.
auto_timestamp

auto_timestamp — adds timestamps to unstamped changes.

Synopsis

(auto_timestamp 0|1)

Arguments

0|1

Use 1 to add timestamps, or 0 to not add timestamps automatically.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command instructs the DataHub to timestamp points automatically to changes that don’t already have a timestamp. This corresponds to the Automatically add a timestamp to unstamped changes checkbox in the General option of the Properties window.
**bridge**

*bridge* — creates a bridge between two points.

**Synopsis**

```c
(bridge source destination
 [flags [multiply add [srcmin srcmax dstmin dstmax]]])
```

**Arguments**

**source**

The fully qualified point name of the source, or starting point of the bridge.

**destination**

The fully qualified point name of the destination, or ending point of the bridge.

**flags**

A bitwise combination of:

- 1: Forward bridge: bridge from source to destination
- 2: Inverse bridge: bridge from destination to source
- 16: Clamp output to the minimum. (Range mapping only.)
- 32: Clamp output to the maximum. (Range mapping only.)
- 256: The bridge is a direct copy
- 512: The bridge uses a linear transformation
- 1024: The bridge uses range mapping
- 4096: The bridge is disabled

**Note:** Bits 256, 512 and 1024 are mutually exclusive.

**multiply**

The multiplier value for a linear transformation. This is ignored if `(flags & 512) == 0`.

**add**

The adder value for a linear transformation. This is ignored if `(flags & 512) == 0`.

**srcmin**

The minimum range map value for the source point. This is ignored if `(flags & 1024) == 0`.

**srcmax**

The maximum range map value for the source point. This is ignored if `(flags &
bridge

1024) == 0.

dstmin
The minimum range map value for the destination point. This is ignored if (flags & 1024) == 0.

dstmax
The maximum range map value for the destination point. This is ignored if (flags & 1024) == 0.

Returns
A message indicating success or error. Please refer to Return Syntax for details.

Description
This command creates a bridge between two data points so that a change to the value of one point automatically propagates to the other point. The scaling and the limits on source and destination points used for linear transformations are stored with the bridge so that if you decide to change from a direct bridge to one that uses linear transformations your previous entries are preserved. The values themselves are only applied when the flag set indicates the corresponding transfer function.
bridge_remove

bridge_remove — deletes a bridge.

Synopsis

```
(bridge_remove source destination)
```

Arguments

- **source**
  A string containing the fully qualified point name of the source, or starting point of the bridge.

- **destination**
  A string containing the fully qualified point name of the destination, or ending point of the bridge.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command permanently deletes a bridge. The source and destination names must be fully qualified, specifying the domain and point name, as in "domain:pointname".
bridge_remove_pattern

bridge_remove_pattern — deletes all bridges that match a pattern.

Synopsis

\{bridge_remove_pattern\} source_pattern destination_pattern

Arguments

source_pattern
A string containing a pattern for the name of the source points of the bridge, such as "domain1:*".

destination_pattern
A string containing a pattern for the name of the destination points of the bridge, such as "*:*".

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command permanently deletes all bridges that match a specified pattern. The available patterns are as follows:

- * matches any number of characters, including zero.
- [c] matches a single character which is a member of the set contained within the square brackets.
- [\c] matches any single character which is not a member of the set contained within the square brackets.
- ? matches a single character.
- {xx,yy} matches either of the simple strings contained within the braces.
- \c (a backslash followed by a character) - matches that character.
bridge_transform

bridge_transform — modifies an existing bridge.

Synopsis

```
{bridge_transform name flags [multiply add [srcmin srcmax dstmin dstmax]]}
```

Arguments

**name**

The name of the bridge.

**flags**

A bitwise combination of:

- 1: Forward bridge: bridge from source to destination
- 2: Inverse bridge: bridge from destination to source
- 16: Clamp output to the minimum. (Range mapping only.)
- 32: Clamp output to the maximum. (Range mapping only.)
- 256: The bridge is a direct copy
- 512: The bridge uses a linear transformation
- 1024: The bridge uses range mapping
- 4096: The bridge is disabled

Bits 256, 512 and 1024 are mutually exclusive.

**multiply**

The multiplier value for a linear transformation. This is ignored if \((flags \& 512) == 0\).

**add**

The adder value for a linear transformation. This is ignored if \((flags \& 512) == 0\).

**srcmin**

The minimum range map value for the source point. This is ignored if \((flags \& 1024) == 0\).

**srcmax**

The maximum range map value for the source point. This is ignored if \((flags \& 1024) == 0\).
**dstmin**

The minimum range map value for the destination point. This is ignored if \((\text{flags} \& 1024) == 0\).

**dstmax**

The maximum range map value for the destination point. This is ignored if \((\text{flags} \& 1024) == 0\).

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command modifies an existing bridge between two data points. The scaling and the limits on source and destination points used for linear transformations are stored with the bridge so that if you decide to change from a direct bridge to one that uses linear transformations your previous entries are preserved. The values themselves are only applied when the flag set indicates the corresponding transfer function.
cforce

cforce — creates a point and forces a value to be written to it.

Synopsis

(cforce name value [confidence])

Arguments

name

   The name of the point. This is a string.

type

   A string representation of the value for the point. It will be interpreted into the type
   specified by the type parameter.

value

   A string representation of the value for the point. The point value will be interpreted
   as integer, float or string based on the contents of the value string. This function tries
   each type in order, and uses the first type for which the value parameter is a valid rep-
   resentation. Double quotes around the value parameter are ignored. For example:

   • 123 is an integer.
   • 123.4 is a float.
   • 1.234e2 is a float.
   • "123" is an integer.
   • 123abc is a string.

   All strings can be surrounded by double-quotes if the string contains spaces or special
   characters. The backslash character (\) escapes double quotes and backslashes within the
   string. Newline, carriage return, form feed and tab are represented with \n, \r, \f, \t re-
   spectively. Strings must not contain newline characters.

Returns

   A message indicating success or error. Please refer to Return Syntax for details.

Description

   This command is the same as cset, except that it forces a write even if the DataHub would
   otherwise refuse it, for example if the point is old, the value is insignificant or hasn't
   changed, or the point is marked as read-only. When this value is set, the following attribut-
   es of the point are set as follows:

   • seconds and nanoseconds are set to the current time on the machine running the
DataHub.

- locked, sec, and quality are all maintained at their previous values for this point.
- flags is set to 0.

Please refer to the **write** command for more information about these parameters. See also **set** and **force**.
cread

cread — creates and reads a point.

Synopsis

(cread name)

Arguments

name
The name of the point.

Returns

The complete point definition in a message, with this syntax:

(point name type value
 [conf security locked seconds nanoseconds flags quality])

where:

name
The name of the point.

type
The data type of the point, one of integer, floating point, or character string.

value
The value of the point.

conf
The confidence level of the point, 0 - 100 percent, unused by most applications.

security
The security level of the point, 0 to 32768, where higher numbers represent higher security.

locked
0 for locked, or 1 for unlocked.

seconds
The operating system time in seconds when the point was read.

nanoseconds
The number of nanoseconds after seconds when the point was read.

flags
User-defined flags.
quality
A constant representing a quality of the connection, assigned by the DataHub for this point, such as Good, Bad, Last known, Local override, etc. The possible values are those supported by OPC in Microsoft Windows.

Description
This command creates a point and then reads the information it contains. See also read.
create

create — creates a new point.

Synopsis

(\texttt{create \textit{name} [0|1]})

Arguments

\textit{name}  
The name of the point, as a string.

0|1  
Tells \texttt{create} what to do if a point already exists with that name. Use 1 to ignore an existing point and do nothing. Use 0 to have \texttt{create} throw an error. If nothing is entered, the default is 0.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a new point in the DataHub. Normally it is not necessary to create points manually—the DataHub creates a new point any time a program sends one. However, this command is useful for creating points programmatically from within the DataHub. See also \texttt{cset}.

Example

Using the Gamma \texttt{datahub\_command} function, you could pass a \texttt{create} command to the DataHub as follows to create \texttt{MyNewPoint} in the \texttt{default} domain, and assign it a value of 1.

\begin{verbatim}
datahub\_command ("(create default:MyNewPoint)", 1);
\end{verbatim}
create_domain

create_domain — creates a new domain.

Synopsis

\{(create_domain name)\}

Arguments

name

The name of the domain, as a string.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a new domain in the DataHub. This corresponds to using the Add button in the Domains section in the General option of the Properties window.
creport

creport — creates a point and requests notification of changes.

Synopsis

(creport name)

Arguments

The name of the data point to be created.

Returns

A message with the complete definition of the point.

Description

This command creates a data point and then requests the DataHub to report changes (also called exceptions) to the value or any other information about the point, as soon as any change takes place. See also report.
cset

cset — creates a point and assigns it a value.

Synopsis

(cset name value [confidence])

Arguments

name
The name of the point. This is a string.

value
A string representation of the value for the point. The point value will be interpreted as integer, float or string based on the contents of the value string. This function tries each type in order, and uses the first type for which the value parameter is a valid representation. Double quotes around the value parameter are ignored. For example:

- 123 is an integer.
- 123.4 is a float.
- 1.234e2 is a float.
- "123" is an integer.
- 123abc is a string.

confidence
A confidence factor in the range of 0 to 100 (optional). This is not used by the DataHub, so is available to programs that produce graduated confidence, such as expert systems. If this value is not specified, it is set to 100.

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (\) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with \n, \r, \f, \t respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This is a convenience command that combines create and set, allowing you to create a point and set its value in a single command. If the point already exists, cset will simply set the value. When this value is set, the following attributes of the point are set as follows:

- seconds and nanoseconds are set to the current time on the machine running the
DataHub.

- locked, sec, and quality are all maintained at their previous values for this point.
- flags is set to 0.

Please refer to the write command for more information about these parameters. See also set, force, and cforce.
cwrite

**cwrite** — creates a point and writes information to it.

**Synopsis**

```
(cwrite name type value conf sec locked seconds nanoseconds
 [flags quality])
```

**Arguments**

**name**

The name of the point. This is a string.

**type**

The type of point. One of
- 0 = string
- 1 = float (8-byte double)
- 2 = integer (32-bit integer)

**value**

A string representation of the value for the point. It will be interpreted into the type specified by the *type* parameter.

**conf**

A confidence factor in the range of 0 to 100. This is not used by the DataHub, so is available to programs that produce graduated confidence, such as expert systems.

**sec**

A security level for this point. This is rarely used. If a point's security level is set to a non-zero value then attempts to write to that point must claim a security level equal to or greater than the security level of the point. This uses a "good citizen" model - the writer can claim any security it wants, and is assumed to be honest - so there is no strong security here. It is intended for systems that want to avoid accidental changes to values. Security level can be from 0 to 32767.

**locked**

An indication that the point is locked, and cannot be changed. Can be 0 or 1. Attempts to write to a locked point will fail.

**seconds**

The UNIX epoch - seconds since Jan. 1, 1970, as produced by the `time()` function.

**nanoseconds**

The number of nanoseconds inside this second. Cannot exceed 1,000,000,000.

**flags**

User level code should always send a 0 for this value.
**quality**

A quality indicator consistent with the OPC DA specification. This is not a bit field. It can be one of:

- **PT_QUALITY_BAD** 0
- **PT_QUALITY_UNCERTAIN** 0x40
- **PT_QUALITY_GOOD** 0xc0
- **PT_QUALITY_CONFIG_ERROR** 0x04
- **PT_QUALITY_NOT_CONNECTED** 0x08
- **PT_QUALITY_DEVICE_FAILURE** 0x0c
- **PT_QUALITY_SENSOR_FAILURE** 0x10
- **PT_QUALITY_LAST_KNOWN** 0x14
- **PT_QUALITY_COMM_FAILURE** 0x18
- **PT_QUALITY_OUT_OF_SERVICE** 0x1c
- **PT_QUALITY_WAITING_FOR_INITIAL_DATA** 0x20
- **PT_QUALITY_LAST_USABLE** 0x44
- **PT_QUALITY_SENSOR_CAL** 0x50
- **PT_QUALITY_EGU_EXCEEDED** 0x54
- **PT_QUALITY_SUB_NORMAL** 0x58
- **PT_QUALITY_LOCAL_OVERRIDE** 0xd8

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (\) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with \n, \r, \f, \t respectively. Strings must not contain newline characters.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command lets you manually write information to a point. If the point does not exist, the point is automatically created and the value is written. This command is identical to the **write** command, except that **write** will produce an error if the point does not exist.
DDEAdvise

DDEAdvise — sets up the item for a DDEAdvise connection.

Synopsis

(DDEAdvise label item pointname [domain])

Arguments

label
A string that identifies the connection for this item.

item
The item name, as a string

pointname
The name of the point with which the item is associated, as a string.

domain
The domain of the point. If unspecified, it defaults to the default domain.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command supports a DDEAdvise connection. It associates a DDE item with a point and domain, as well as the label used by the item’s service and topic (see DDEConnect). This command is used when you add or edit connections in the DDE option of the Properties window. For more information on DDE connections, please see Appendix G, DDE Overview.
DDEConnect

DDEConnect — makes a connection to a DDE service and topic.

Synopsis

\[(DDEConnect \ label \ service \ topic \ [retry\_sec])\]

Arguments

- **label**
  A string that identifies this connection.

- **service**
  The DDE service name, as a string.

- **topic**
  The DDE topic, as a string.

- **retry\_sec**
  The time interval, in seconds, that the connection should be retried in case of a dis-connect or network failure.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command establishes a connection to a DDE service and topic, and names the connection. The point or points that use this connection are defined by the DDEAdvise command. Both of these commands are used when you add or edit connections in the DDE option of the Properties window. For more information on DDE connections, please see Appendix G, DDE Overview.
DDEDisconnect

DDEDisconnect — disconnects and discards a DDE connection.

Synopsis

```
(DDEDisconnect label)
```

Arguments

`label`
A label identifying a connection, as assigned by the DDEConnect command.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command disconnects and discards a DDE connection that had been previously established by the DDEConnect command.
**DDEExcelUnicode**

*DDEExcelUnicode* — accepts Unicode characters in strings in Excel.

**Synopsis**

```
(DDEExcelUnicode 0|1)
```

**Arguments**

0|1

1 enables the capability, 0 disables it.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command enables or disables the capability of the DataHub to function properly with Unicode characters in strings in Excel. This corresponds to the **Accept non-English characters in Excel strings (slower)** checkbox in the **DDE option** of the Properties window.>
DDEInit

**DDEInit** — initializes the DataHub to act as a DDE server.

**Synopsis**

```
(DDEInit)
```

**Arguments**

none

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command instructs the DataHub to act as a DDE server. Service names are assigned using the **DDEService** command.
DDEService

**Synopsis**

(DDEService service)

**Arguments**

*service*

A DDE service name for the DataHub, as a string.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command assigns a DDE service name to the DataHub. It is possible to use several names simultaneously. This command is used when adding service names to the DataHub in the DDE option of the Properties window. Also see DDEInit.
DDEUnadvise

**DDEUnadvise** — removes an item from a DDE connection.

**Synopsis**

```
(DDEUnadvise label item)
```

**Arguments**

- `label`
  A label identifying a connection, as assigned by the **DDEConnect** command.

- `item`
  The DDE item name of the point you wish to stop advising on.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command essentially undoes the **DDEAdvise** command, removing an item from a DDE connection.
DDEUnadvisePattern

**Synopsis**

\[(DDEUnadvisePattern \ label \ pattern)\]

**Arguments**

- **label**
  A label identifying a connection, as assigned by the DDEConnect command.

- **pattern**
  A pattern that matches the DDE item names of the points you wish to stop advising on.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command undoes the DDEAdvise command for a group of points that match the pattern, removing those items from a DDE connection. The available patterns are as follows:

- * matches any number of characters, including zero.
- [c] matches a single character which is a member of the set contained within the square brackets.
- [\c] matches any single character which is not a member of the set contained within the square brackets.
- ? matches a single character.
- {xx,yy} matches either of the simple strings contained within the braces.
- \c (a backslash followed by a character) - matches that character.
DDEUnadvisePoint

DDEUnadvisePoint — removes an item from a DDE connection, by its point name.

Synopsis

(DDEUnadvisePoint label pointname)

Arguments

label
A label identifying a connection, as assigned by the DDEConnect.

point
The point name corresponding to the DDE item you wish to stop advising on.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command essentially undoes the DDEAdvise command, removing an item from a DDE connection.
debug

ddebug — sets the debug level.

Synopsis

(debug debug_level)

Arguments

debug_level
An integer from 0 to 4 specifying the debug level.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you specify the level of detail of debugging, from the least (0) to the most (4).
defaultprop

**defaultprop** — sets a default type for a property.

**Synopsis**

```
(defaultprop domain type propname)
```

**Arguments**

- **domain**
  The domain name of the property whose default type will be set.

- **type**
  The default type for this property, as a string.

- **propname**
  The name of the property, as a string.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command sets a default type for a property on a given domain. For more information and an example, please refer to the section called “Assemblies, Subassemblies, Attributes, and Properties”.
delete

**Synopsis**

```plaintext
(delete pointname [domain])
```

**Arguments**

*pointname*

The name of the point to delete, as a string.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

Deleting points with this command could cause unexpected behavior for other users of the point.

This command allows you to delete points from the DataHub. You should exercise caution when using it, however, because any program connected to the DataHub now or in the future might be relying on the existence of that point. We suggest you use this command only after ensuring that no connecting program uses that point.
**div**

_div_ — does division on the value of a point.

**Synopsis**

```
(append name number [secs] [nano])
```

**Arguments**

- `name`:
  The name of a point, which must be a number type.

- `number`:
  A value to divide the value of the point by.

- `secs`:
  The time in seconds. If this is not specified, the current date and time in seconds is used.

- `nano`:
  A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command is used to divide to the value of a point.
domain

domain — identifies the client domain name.

Synopsis

(domain domain_name)

Arguments

domain_name

The name of the domain.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command identifies to the DataHub the domain that the client is using.
domain_bridge

domain_bridge — configures a redundancy set.

Synopsis

(domain_bridge label src1 src2 dst flags switchflags point quality value statepoint controlpoint statuspoint1 statuspoint2 [fill_delay_ms])

Arguments

Many of these parameters correspond to a similarly named entry field or checkbox in the Configure Redundancy dialog options in the Redundancy option of the Properties window. The names that appear in that dialog are shown here in parentheses.

label
A label for this redundancy set, as a string. (Label:)

src1
The domain for the first data source, as a string. (Source Domain 1:)

src2
The domain for the second data source, as a string. (Source Domain 2:)

dst
The name of the domain that will contain the output. (Output Domain)

flags
are not yet documented.

switchflags
are not yet documented.

point
The name of a point which will be used in comparisons. (For this point)

quality
The quality of the data which will be used in comparisons, as a number. The valid numbers and corresponding qualities can be found in quality. (Data quality is:)

value
The value of the data which will be used in comparisons, as a number. (Data value is:)

statepoint
The name of a point that will be used to indicate which data source is currently being used. (Point for current source number:)

controlpoint
The name of a point that will be used to indicate which data source is preferred.
(Point for preferred source number:)

`statuspoint1`

The name of a point that will be used to indicate the current status of the first data domain. *(Point for the current state of domain 1:)*

`statuspoint2`

The name of a point that will be used to indicate the current status of the second data domain. *(Point for the current state of domain 2:)*

`fill_delay_ms`

The number of seconds to wait before switching domains if the data flow detection is activated. *(Switch sources if data stops for:)*

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command configures a DataHub redundancy set. It corresponds to the **Configure Redundancy** dialog options in the **Redundancy** option of the Properties window.
domain_bridge_enable

domain_bridge_enable — enables or disables the Redundancy feature.

Synopsis

\{(domain_bridge_enable label [enabled])\}

Arguments

label
  The label of the redundant connection, as seen in the in the Redundancy option of the Properties window.

enabled
  A value of 1 enables the capability, 0 disables it. The default value is 1.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the DataHub Redundancy feature. It corresponds to the Redundancy checkbox in the Redundancy option of the Properties window.
domain_bridge_prefer

domain_bridge_prefer — specifies a preferred source for a Redundancy connection.

Synopsis

\{(domain_bridge_prefer label [index])\}

Arguments

label
  The label of the redundant connection, as seen in the in the Redundancy option of the Properties window.

index
  a value of 1 specifies Source Domain 1, while a value of 2 specifies Source Domain 2. The default value is 1.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command specifies a preferred source for a Redundancy connection. It corresponds to the Preferred source checkboxes in the Configure Redundancy option of the Properties window.
domain_bridge_refresh

domain_bridge_refresh — refreshes a Redundancy connection.

Synopsis

(domain_bridge_refresh label)

Arguments

label

The label of the redundant connection, as seen in the Redundancy option of the Properties window.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command refreshes a Redundancy connection, typically after reconfiguration.
**domain_bridge_remove**

*domain_bridge_remove* — removes a Redundancy connection.

**Synopsis**

```plaintext
(domain_bridge_remove label)
```

**Arguments**

*label*

The label of the redundant connection to remove, as seen in the Redundancy option of the Properties window.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command removes a configured Redundancy connection. It corresponds to clicking the *Remove* button for the selected label in the Redundancy option of the Properties window.
domains

domains — lists all domains in the DataHub.

Synopsis

(domains)

Arguments

none

Returns

On success, a message with the following syntax:

(domains "default" "domain1" "domain2" "domain3" ...)

the following syntax. On error, an error message as described in Return Syntax.

Description

This command generates a response message listing all of the domains currently in the DataHub. The response message is different from the usual DataHub command return syntax, in that it doesn't contain the string success on success.
dump

dump — writes the entire content of the DataHub to a file.

Synopsis

(dump filename)

Arguments

filename
The name of the file to write to.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command dumps the contents of the DataHub to a file. The results are listed by domain, data types, and assemblies.

Example

You can create a dump of the DataHub's current point list by issuing this command in the Script Log entry field:

datahub_command("(dump \"c:/temp/datahub.dump\")", 1)
enable_bridging

enable_bridging — enables or disables bridging capabilities.

Synopsis

\{(enable_bridging 0|1)\}

Arguments

0|1

1 enables the capability, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the DataHub to support bridging.

See Also

bridge
enable_dde_client

enable_dde_client — enables or disables DDE client capabilities.

Synopsis

```
(enable_dde_client 0|1)
```

Arguments

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 enables the capability, 0 disables it.</td>
</tr>
</tbody>
</table>

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the DataHub to act as a DDE client. This corresponds to the Act as a DDE client... checkbox in the DDE option of the Properties window.
**enable_dde_server**

**enable_dde_server** — enables or disables DDE server capabilities.

**Synopsis**

```bash
(enable_dde_server 0|1)
```

**Arguments**

0 | 1
--- | ---
1 enables the capability, 0 disables it.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command enables or disables the capability of the DataHub to act as a DDE server. This corresponds to the Act as a DDE server... checkbox in the DDE option of the Properties window.
enable_domain_bridging

Synopsis

```
(enable_domain_bridging 0|1)
```

Arguments

0|1

1 enables the capability, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command has not yet been documented.
enable_opc_client

enable_opc_client — enables or disables OPC client capabilities.

Synopsis

```
(enable_opc_client 0|1)
```

Arguments

0|1

1 enables the capability, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the DataHub to act as an OPC client. This corresponds to the Act as an OPC Client... checkbox in the OPC option of the Properties window.
enable_opc_server

enable_opc_server — enables or disables OPC server capabilities.

Synopsis

```
(enable_opc_server 0|1)
```

Arguments

0 | 1
1 enables the capability, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the DataHub to act as an OPC server. This corresponds to the Act as an OPC Server. checkbox in the OPC option of the Properties window.
enable_scripting

enable_scripting — enables or disables scripting capabilities

Synopsis

(enable_scripting 0|1)

Arguments

0|1

1 enables the capability, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the DataHub to support scripting.
**error**

`error` — sends an error with an error string.

**Synopsis**

```plaintext
(error errstring)
```

**Arguments**

`errstring`

- a string containing an error message.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command sends an error to the DataHub, which it displays in the Event Log, along with the message from the `errstring`. 
execute_plugin

execute_plugin — executes a plugin. (*experimental*)

Synopsis

{execute_plugin plugin_name}

Arguments

plugin_name

The name of a plugin.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the DataHub to execute a plugin.

The commands related to plugins are currently experimental.
exit

exit — shuts down the DataHub.

Synopsis

{exit [exit_status]}

Arguments

exit_status
An optional string to describe the circumstances of the shut-down.

Returns

A message containing the exit_status message.

Description

This command closes the DataHub and all associated windows.
force

force — forces a write to a point.

Synopsis

(force name value [confidence])

Arguments

name

The name of the point. This is a string.

value

A string representation of the value for the point. The point value will be interpreted
as integer, float or string based on the contents of the value string. This function tries
each type in order, and uses the first type for which the value parameter is a valid rep-
resentation. Double quotes around the value parameter are ignored. For example:

• 123 is an integer.
• 123.4 is a float.
• 1.234e2 is a float.
• "123" is an integer.
• 123abc is a string.

certainty

A confidence factor in the range of 0 to 100 (optional). This is not used by the
DataHub, so is available to programs that produce graduated confidence, such as ex-
pert systems. If this value is not specified, it is set to 100.

All strings can be surrounded by double-quotes if the string contains spaces or special
characters. The backslash character (\) escapes double quotes and backslashes within the
string. Newline, carriage return, form feed and tab are represented with \n, \r, \f, \t re-
spectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is the same as set, except that it forces a write even if the DataHub would
otherwise refuse it, for example if the point is old, the value is insignificant or hasn’t
changed, or the point is marked as read-only. When this value is set, the following attribut-
es of the point are set as follows:
• seconds and nanoseconds are set to the current time on the machine running the DataHub.
• locked, sec, and quality are all maintained at their previous values for this point.
• flags is set to 0.

Please refer to the write command for more information about these parameters. See also set and cforce.
format

format — is an efficiency enhancement for Linux.

Synopsis

(format flag)

Arguments

flag

One of the following flags:

• 1 turns on the behavior.
• 0 turns off the behavior.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is an efficiency enhancement for Linux, for SRR Module connections only. It tells the DataHub that this client would like to get point exceptions in binary instead of ASCII encoding. It is more CPU efficient but does not work well in a heterogeneous network.
**get_client_stats**

`get_client_stats` — provides statistics for all clients.

**Synopsis**

```
(get_client_stats [label_pattern])
```

**Arguments**

`label_pattern`

Not yet documented.

**Returns**

A list containing each DataHub client and its associated statistics.

**Description**

This command generates a list of DataHub clients, along with their statistics in Lisp format. The statistics in the list correspond to the statistics displayed in the Connection Viewer.

**Example**

```lisp
---> pretty_princ(datahub_command ("(get_client_stats)", 1));
("get_client_stats" (  
("" "Gamma Scripting Engine" "Gamma Scripting Engine" "" " " "
  ((PointsIn 0)(PointsOut 4)(ConnectOk 0)(ConnectFail 0)
   (Disconnect 0)(Block 0)(Unblock 0)(PointsCreate 0)
   (PointsRead 0)(PointsRegister 2)(PointsUnregister 0)
   (CmdInvalid 0)(CmdFailed 0)(PointsDropped 0)(QueueSize 0)))
("" "TCP Outgoing" "Outgoing plain text to developers.cogentrts.com:4502 into domain test" "Running" "" "
  ((PointsIn 88495)(PointsOut 0)(ConnectOk 0)(ConnectFail 0)
   (Disconnect 0)(Block 0)(Unblock 0)(PointsCreate 11)
   (PointsRead 0)(PointsRegister 0)(PointsUnregister 0)
   (CmdInvalid 0)(CmdFailed 0)(PointsDropped 0)(QueueSize 0)))
("" "Mainline" "Mainline" "" " "
  ((PointsIn 0)(PointsOut 0)(ConnectOk 0)(ConnectFail 0)
   (Disconnect 0)(Block 0)(Unblock 0)(PointsCreate 4)
   (PointsRead 0)(PointsRegister 0)(PointsUnregister 0)
   (CmdInvalid 0)(CmdFailed 0)(PointsDropped 0)(QueueSize 0)))
("" "OPCAE" "OPC A&E Server: 0 client connections" " " " "
  ((PointsIn 0)(PointsOut 9)(ConnectOk 0)(ConnectFail 0)
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
get_client_stats

(Disconnect 0) (Block 0) (Unblock 0) (PointsCreate 0) (PointsRead 0) (PointsRegister 16) (PointsUnregister 0) (CmdInvalid 0) (CmdFailed 0) (PointsDropped 0) (QueueSize 0))
heartbeat

**heartbeat** — establishes a heartbeat message.

**Synopsis**

```
(heartbeat ms)
```

**Arguments**

*ms*

The number of milliseconds between each pulse.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command establishes a heartbeat message that verifies the connection every *ms* milliseconds. The heartbeat is sent from the DataHub to the client.
HistorianAdd

**HistorianAdd** — adds a point to a Historian group.

**Synopsis**

```
(HistorianAdd point [grouplabel])
```

**Arguments**

- **point**
  - The name of the point to be added.
- **grouplabel**
  - The name of the Historian group to which the point will be added, such as HIST000.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command adds a point to the specified Historian group, which corresponds to adding a point using the Configure Historical Data Capture interface in the Historian option of the Properties window.
HistorianFlags

HistorianFlags — is new, not yet documented.

Synopsis

\{(HistorianFlags flags)\}

Description

This command has not yet been documented.
**HistorianRemove**

*HistorianRemove* — removes points.

**Synopsis**

```plaintext
(HistorianRemove point_pattern [group_pattern])
```

**Arguments**

- `point_pattern`  
  A pattern that matches one or more names of the points to be removed.

- `group_pattern`  
  A pattern that matches names of Historian groups, to select the groups from which points will be removed, such as HIST0.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command deletes histories for selected points in selected groups. It corresponds to unchecking the box associated with the `point_pattern` in the Historian option of the Properties window.
HistorianSaveConfig

**HistorianSaveConfig** — saves the configuration of the Historian.

**Synopsis**

```plaintext
(HistorianSaveConfig)
```

**Arguments**

none

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command saves the configuration of the Historian.
**HistorianSetConfiguring**

*HistorianSetConfiguring* — sets the Historian status to "Configuring".

**Synopsis**

```
(HistorianSetConfiguring 0|1)
```

**Arguments**

0|1

1 sets the Historian status to "Configuring", 0 releases it.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command sets the Historian status to "Configuring", or releases it. There is no corresponding item in the Historian option of the Properties window for this.
HistoryGroupAdd

HistoryGroupAdd — adds a new history group.

Synopsis

```
(HistoryGroupAdd group_label basedir extension cachesize flags)
```

Arguments

- `group_label`:
  The label for the group, such as HIST000.

- `basedir`:
  The name of the base directory for the group, where history files will be stored.

- `extension`:
  A filename extension for the history files.

- `cachesize`:
  The size of the in-memory cache, which can have an impact on the speed of data recall.

- `flags`:
  Not yet documented.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command adds a new history group. Its parameters correspond to the Group Configuration options in the Configure Historical Data Capture dialog of the Historian option of the Properties window.
HistoryGroupAddPoint

**HistoryGroupAddPoint** — adds a point to a Historian group.

**Synopsis**

```
(HistoryGroupAddPoint  group_label  pointname)
```

**Arguments**

- `group_label`
  - The label for the group, such as HIST000.
- `pointname`
  - The name of the point to be added.

**Returns**

A message indicating success or error. Please refer to [Return Syntax](#) for details.

**Description**

This command adds a point to a Historian group. This corresponds to adding points in the Configure Historical Data Capture dialog of the Historian option of the Properties window.
HistoryGroupDeadband

**HistoryGroupDeadband** — sets a deadband for a Historian group.

**Synopsis**

```plaintext
(HistoryGroupDeadband group_pattern flags absolute percent maxsecs maxcount)
```

**Arguments**

- **group_pattern**
  A pattern to match one or more groups, such as `HIST`.
- **flags**
  Not yet documented.
- **absolute**
  Sets the deadband range to a single value.
- **percent**
  Sets the deadband range to be a percent of the last logged data value.
- **maxsecs**
  Sets the maximum time period (in seconds, a real number) to deadband.
- **maxcount**
  Sets a maximum number of values to skip for the deadband.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command sets a deadband for a Historian group. This corresponds to the Deadband options in the Historian option of the Properties window. Please refer to that section of the documentation for more details about the deadband options.
HistoryGroupDefault

HistoryGroupDefault — sets a default group.

Synopsis

```
{HistoryGroupDefault label}
```

Arguments

`label`

The label for the group, such as HIST000.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets a default group, to which all new requests are automatically assigned, if not specified otherwise. This corresponds to checking the Automatically assign new requests to this group checkbox in the Historian option of the Properties window.
**HistoryGroupFileTimes**

*HistoryGroupFileTimes* — sets a frequency for changing history files.

**Synopsis**

```
(HistoryGroupFileTimes group_pattern days hours minutes)
```

**Arguments**

- **group_pattern**
  - A pattern to match one or more groups, such as HIST.

- **days**
  - The number of days between file changes. Enter 0 to ignore this parameter.

- **hours**
  - The number of hours between file changes. Enter 0 to ignore this parameter.

- **minutes**
  - The number of minutes between file changes. Enter 0 to ignore this parameter.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command sets a frequency for changing history files. This corresponds to the *Change file every* option in the Configure Historical Data Capture dialog of the Historian option of the Properties window.
HistoryGroupFlushTimes

**HistoryGroupFlushTimes** — specifies how frequently to flush history data to disk.

**Synopsis**

```
(HistoryGroupFlushTimes group_pattern days hours minutes)
```

**Arguments**

- **group_pattern**
  A pattern to match one or more groups, such as HIST.
- **days**
  The number of days between disk writes. Enter 0 to ignore this parameter.
- **hours**
  The number of hours between disk writes. Enter 0 to ignore this parameter.
- **minutes**
  The number of minutes between disk writes. Enter 0 to ignore this parameter.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command specifies how frequently to flush history data to disk. This corresponds to the **Flush to disk every** option in the Configure Historical Data Capture dialog of the Historian option of the Properties window.
**HistoryGroupRemove**

**HistoryGroupRemove** — removes a group and all of its histories.

**Synopsis**

```
(HistoryGroupRemove group_pattern)
```

**Arguments**

*group_pattern*

A pattern to match one or more groups, such as `HIST`.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command removes a group and all of its histories. This corresponds to the **Remove** button in the Historian option of the Properties window.
HistoryGroupStorageTimes

**HistoryGroupStorageTimes** — specifies how long to keep data in a history.

**Synopsis**

```
(HistoryGroupStorageTimes group_pattern days hours minutes)
```

**Arguments**

- **group_pattern**
  - A pattern to match one or more groups, such as HIST.
- **days**
  - The number of days to store data. Enter 0 to ignore this parameter.
- **hours**
  - The number of hours to store data. Enter 0 to ignore this parameter.
- **minutes**
  - The number of minutes to store data. Enter 0 to ignore this parameter.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command specifies how long to keep data in a history. This corresponds to the **Keep data for** option in the Configure Historical Data Capture dialog of the Historian option of the Properties window.
ignore

**ignore** — ignores a given point.

**Synopsis**

```plaintext
(ignore pointname)
```

**Arguments**

`pointname`

The name of the point to be ignored.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command tells the DataHub to ignore changes in value to the point `pointname`. 
ignore_old_data

ignore_old_data — ignores changes with an old timestamp.

Synopsis

(ignore_old_data 0|1)

Arguments

0|1

Use 1 to ignore old data, or 0 to not ignore old data.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command instructs the DataHub to ignore any changes to data that arrive with an old timestamp. This corresponds to the Do not transmit changes with an old timestamp checkbox in the General option of the Properties window.
**include**

`include` — includes a file in with configuration files.

**Synopsis**

```
{include filename [enabled] [as-sender-0|1]}
```

**Arguments**

- **filename**
  The name of the configuration file to include.
- **enabled**
  One of the following flags indicating the enabled state.
  - 1 means enabled; it will be used immediately after loading.
  - 0 means disabled; it will be loaded but not used.
- **as-sender-0|1**
  Not yet documented.

**Returns**

A message indicating success or error. Please refer to *Return Syntax* for details.

**Description**

This command includes a file with configuration files. It can be used in a configuration file itself, causing the included file to be inserted into the configuration file at the point of this command. Or it can be sent to the DataHub by another program. The *enabled* parameter puts the file into the enabled state, meaning that it is loaded and then immediately used. With that parameter turned off, the file is simply added to the DataHub's list of configuration files.
instance

instance — creates an instance of a data organization model.

Synopsis

(instance domain pointname assemblyname)

Arguments

domain
    The domain of the model.
pointname
    A name for the instance.
assemblyname
    The assembly associated with the model.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates an instance of a data organization model, based on a given domain and assembly. For more information and an example, please refer to the section called “Data Organization”.
load_config_files

**Synopsis**

```
(load_config_files 0|1)
```

**Arguments**

0 | 1

Use 1 to have configuration files loaded, or 0 to not have them loaded.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command lets you instruct the DataHub to load or not load its configuration files.
load_plugin

load_plugin — loads a specified plugin. (experimental)

Synopsis

(load_plugin plugin_name run_now)

Arguments

plugin_name
The name of a plugin.

run_now
One of the following flags:
• 1 means the plugin will run immediately after it is loaded.
• 0 means disabled; the plugin will be loaded but not run.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the DataHub to load a plugin.

The commands related to plugins are currently experimental.
load_scripts

load_scripts — loads scripts.

Synopsis

(load_scripts filename enabled)

Arguments

filename
The name of a script.

enabled
One of the following flags indicating the enabled state.

• 1 means enabled; the script will be run immediately after it is loaded.
• 0 means disabled; the script will be loaded but not run.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the DataHub to load or not load scripts.
lock

lock — locks and unlocks points.

Synopsis

(lock name secur 0|1)

Arguments

name
The point name.

secur
The security level of the point.

0|1
Use 1 to lock the point, or 0 to unlock the point.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command locks and unlocks points. When a point is locked, it cannot be changed until it is unlocked.
log_file

log_file — sets up a log file.

Synopsis

(log_file filename)

Arguments

logfile

The name of the log file.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets up a log file for messages that appear in the Event Log. If a log file already exists, this command changes that file's name. Starting and stopping logging is done with the log_to_file command. If logging to a file is enabled, the log messages will also continue to appear in the Event Log.
log_file_max

log_file_max — sets a size limit for log files.

Synopsis

(log_file_max  kbytes)

Arguments

kbytes
The maximum file size, in kilobytes.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets a size limit for log files, and corresponds to the Limit entry field in the Event Log window.
log_to_file

Synopsis

(log_to_file 0|1)

Arguments

Use 1 to start logging to a file, or 0 to stop.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command starts or stops logging error messages to the log file set up by log_file.
mirror_master

mirror_master — sets up a mirroring master.

Synopsis

{mirror_master host service localdomain [masterdomain]}

Arguments

host
The master host's name or IP address.

service
The service name or port number of the mirroring master.

localdomain
The domain of the slave (i.e. local) computer

masterdomain
The domain of the mirroring master.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is to be sent to the slave DataHub in a mirroring relationship. It tells the slave all the information it needs about its connection to the master DataHub. This command corresponds to all the entries in the Mirror Master dialog that opens when you click the Add button in the Tunnel/Mirror option of the Properties window.
mirror_master_2

**mirror_master_2** — sets up a secure mirroring master.

**Synopsis**

```
(mirror_master_2 host port flags localdomain remotedomain
  heartbeat timeout [username password])
```

**Arguments**

*host*

The master host's name or IP address.

*port*

The port number or service name of the mirroring master.

*flags*

Any combination of:

<table>
<thead>
<tr>
<th>Hex</th>
<th>Flag</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000001</td>
<td>USE_SSL</td>
<td>Use an SSL connection.</td>
</tr>
<tr>
<td>0x00000002</td>
<td>USE_WEBSOCKET</td>
<td>Use a WebSocket connection.</td>
</tr>
<tr>
<td>0x00000004</td>
<td>VALIDATE_SSL</td>
<td>Validate the SSL certificate when making a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>connection.</td>
</tr>
<tr>
<td>0x00000008</td>
<td>VALIDATE_HOST</td>
<td>Validate the SSL certificate against the host</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name when making a connection.</td>
</tr>
<tr>
<td>0x00000010</td>
<td>USE_PROXY</td>
<td>Use a forward proxy when connecting via</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WebSocket.</td>
</tr>
<tr>
<td>0x00000020</td>
<td>USE_HTTP_CONNECT</td>
<td>Use HTTP CONNECT when connecting to a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>forward proxy.</td>
</tr>
<tr>
<td>0x00000800</td>
<td>DISABLED</td>
<td>Client is disabled (and not connected).</td>
</tr>
<tr>
<td>0x00010000</td>
<td>USE_BINARY</td>
<td>Send point messages in binary rather than ASCII.</td>
</tr>
<tr>
<td>0x00020000</td>
<td>EMBEDDED</td>
<td>Target is an Embedded Toolkit server (ETK).</td>
</tr>
<tr>
<td>0x00100000</td>
<td>READABLE</td>
<td>Data is readable from the master. Always set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>this.</td>
</tr>
<tr>
<td>0x00200000</td>
<td>WRITABLE</td>
<td>Data is writable to the master.</td>
</tr>
<tr>
<td>0x01000000</td>
<td>SYNC_SEND</td>
<td>On initial connection, send all data to the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>master</td>
</tr>
<tr>
<td>0x02000000</td>
<td>SYNC_RECV</td>
<td>On initial connection, receive all data from</td>
</tr>
</tbody>
</table>
### Notes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Flag</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000000</td>
<td></td>
<td>the master</td>
</tr>
<tr>
<td>0x04000000</td>
<td>SYNC_TIME</td>
<td>On initial connection, synchronize by comparing time stamps.</td>
</tr>
<tr>
<td>0x10000000</td>
<td>AUTHORITATIVE</td>
<td>Master is authoritative. Data here is marked Not Connected when the connection drops.</td>
</tr>
<tr>
<td>0x20000000</td>
<td>NON_AUTHORITATIVE</td>
<td>I am authoritative. Data in the master is marked Not Connected when the connection drops.</td>
</tr>
<tr>
<td>0x40000000</td>
<td>OVERRIDE_TIME</td>
<td>Override the time stamp on incoming data with the system clock time.</td>
</tr>
</tbody>
</table>

Some combinations of flags will generate strange results:

- Combining WRITABLE with OVERRIDE_TIME will result in pollution of the master's time stamps.
- Only one of SYNC_SEND, SYNC_RECV and SYNC_TIME should be specified.
- Only one of AUTHORITATIVE and NON_AUTHORITATIVE should be specified.
- If WRITABLE is not set, then SYNC_SEND, SYNC_TIME and NON_AUTHORITATIVE make no sense.

**localdomain**

The domain of the slave (i.e. local) computer

**remotedomain**

The domain of the mirroring master.

**heartbeat**

The number of milliseconds for the heartbeat on this connection. 0 means disabled.

**timeout**

The number of milliseconds for the timeout on this connection. 0 means disabled. The timeout should be significantly longer than the heartbeat.

**username**

(optional) The user name to use if authenticating this connection.

**password**

(optional) The password to use if authenticating this connection. This password is stored in obfuscated text to stop somebody from casually copying it. However, since this obfuscation must be reversible to give access to the original password, it does not represent strong security. Do not allow untrusted people access to your configuration file.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.
Description

This command is the same as mirror_master, but includes the necessary parameters for security. This command is sent to the slave DataHub in a mirroring relationship. It tells the slave all the information it needs about its connection to the master DataHub. This command corresponds to all the entries in the Mirror Master dialog that opens when you click the Add button in the Tunnel/Mirror option of the Properties window.

The mirror_master_2 command was originally used for setting up tunnel connections in the Cogent DataHub.cfg file. When the tunnel code was moved to a separate plug-in (plugin_TCP.cfg), the command format was changed, and renamed master. Now when the DataHub receives a mirror_master_2 command, it converts it into a master command and forwards it to plugin_TCP.cfg.

The master command is not accessible outside of plugin_TCP.cfg. Its syntax is:

```
(master primaryhost primaryport secondaryhost secondaryport
 localdomain remotedomain remoteusername remotepassword flags timeout
 heartbeat retrydelay proxyaddress proxyport proxyusername
 proxypassword)
```

Optional entries that get left blank during configuration are designated by empty quotes ("").
ModbusApplySettings

ModbusApplySettings — applies all scripted changes.

Synopsis

(ModbusApplySettings)

Arguments

None.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used in a script to apply all changes made to Modbus connections. When reconfiguring, you might need to add or remove points, deadbands, etc. These changes don't take effect immediately, they reside temporarily, in memory only. Using the ModbusApplySettings command applies your changes. This command affects all edits since the last time you used it, and is roughly equivalent to clicking the Apply button in the Modbus option of the Properties window.
ModbusCancelSettings

ModbusCancelSettings — cancels all scripted changes.

Synopsis

(ModbusCancelSettings)

Arguments

None.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used in a script to cancel all changes made to Modbus connections. When reconfiguring, you might need to add or remove points, deadbands, etc. These changes don't take effect immediately, they reside temporarily, in memory only. Using the ModbusApplySettings command cancels your changes. This command affects all edits since the last time you used the ModbusApplySettings, and is roughly equivalent to clicking the Cancel button in the Modbus option of the Properties window.
ModbusCreateSlave

**ModbusCreateSlave** — creates a slave connection.

**Synopsis**

```c
(ModbusCreateSlave slaveName hostSpec dataDomain pollMs retryMs maxMsgLength slaveId supportedFunctions addressFlags)
```

**Arguments**

`slaveName`
- Any alphanumeric string identifying this connection.

`hostSpec`
- A list of `(hostType, hostName, port)` where `hostType` should be "tcp"

`dataDomain`
- The data domain into which to write data points.

`pollMs`
- The number of milliseconds in the polling cycle, with a minimum of 10.

`retryMs`
- The number of milliseconds to wait before retrying after a disconnection, with a minimum of 10.

`maxMsgLength`
- The maximum length of a message, in bytes, between 20 and 260.

`slaveId`
- The ID for the slave device. This should match the configuration in the slave.

`supportedFunctions`
- A list of zero or more of (5, 6, 15, 16, 22).

`addressFlags`
- A list of zero or more of ("BitOrderReversed", "OneBasedAddressing", "OneBasedBitAddressing").

**Returns**

A message indicating success or error. Please refer to **Return Syntax** for details.

**Description**

This command creates a slave connection, as specified by the parameters.
ModbusDeleteSlave

ModbusDeleteSlave — deletes a slave connection.

Synopsis

(\text{ModbusDeleteSlave} \text{slaveNamePattern})

Arguments

\textit{slaveNamePattern}  
An alphanumeric string that matches the name(s) of one or more slave connections.

Returns

A message indicating success or error. Please refer to \textit{Return Syntax} for details.

Description

This command deletes one or more slave connections whose names match the pattern specified. This is equivalent to clicking the \textbf{Remove Slave} button for items the list of slave connections in the \textit{Modbus} option of the Properties window.
ModbusEnableMaster

ModbusEnableMaster — enables and disables Modbus master functionality.

Synopsis

(ModbusEnableMaster 0|1)

Arguments

0|1

If 1, the DataHub's Modbus master functionality is enabled; if 0, it is disabled.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

If the argument is 1, the DataHub's Modbus master functionality is enabled. This is equivalent to checking the Enable Modbus master check-box in the Modbus option of the Properties window.
ModbusEnableSlave

ModbusEnableSlave — enables and disables Modbus slave connections.

Synopsis

(ModbusEnableSlave slaveNamePattern 0|1)

Arguments

slaveNamePattern

An alphanumeric string that matches the name(s) of one or more slave connections.

0|1

If 1, the specified Modbus slave connection or connections are enabled; if 0, they are disabled.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables slave connections, the equivalent to the check-boxes in the list of Modbus slave connections in the Modbus option of the Properties window.
ModbusQuerySlave

ModbusQuerySlave — checks for the existence of a slave connection.

Synopsis

(ModbusQuerySlave name)

Arguments

name

The name of a Modbus slave connection, as a string.

Returns

t on success, nil on failure, otherwise an error.

Description

This command checks to see if a given Modbus slave connection exists.
ModbusReloadSettings

ModbusReloadSettings — loads all Modbus configuration.

Synopsis

```
(ModbusReloadSettings)
```

Arguments

None.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command loads all Modbus configuration, including from outside sources, such as from a spreadsheet or a DataHub script. This command is roughly equivalent to clicking the Load Modbus Configuration... button in the Modbus option of the Properties window.
ModbusSlaveAddPoint

ModbusSlaveAddPoint — configures a Modbus point.

Synopsis

(ModbusSlaveAddPoint slaveName pointName blockType pointType address allowWrite xformType xformArgs conversion)

Arguments

slaveName
The name of the Modbus slave, as a string.

pointName
The point name, as a string, not including the data domain name.

blockType
A string, one of
• MB_DI for digital input
• MB_DO for digital output
• MB_AI for analog input
• MB_AO for analog output

pointType
The type of point, as a string, consisting of typesize.flags, where:
• type is one of:
  • i for integer
  • r for real (floating point)
  • s for string
  • b for bit
• size is one of
  • 1 for one bit (b) or one byte (s)
  • 2 for two bytes (i or s)
  • 4 for four bytes (i or r)
  • 8 for eight bytes (i or r)
• flags is any combination of
  • – (a minus sign) to indicate a signed integer (i)
  • b to swap bytes within words for 2, 4, or 8 byte sizes
  • w to swap words within dwords for 4 or 8 byte sizes
• d to swap dwords within quadwords for 8 byte sizes
• u to indicate UTF-8 for string (s) types of size 1, otherwise ASCII. s2 strings are always UTF-16.

For example, "i4.-b" would be a signed, 4-bit integer with bytes swapped within words.

address
The Modbus address, as a string, consisting of one of the following:
• number - the Modbus address offset, starting from 0 or 1 depending on the slave definition
• number.bit - a single bit within the integer starting at address number
• number.bit-bit - a bit field consisting of all bits between the first and last, inclusive
• number[length] - an array, starting at address number

allowWrite
0 for not writable, 1 for writable

xformType
The type of transformation to make, one of: direct, linear or range

For more information about transformations, please refer to the Transform section of the Modbus option of the Properties window.

xformArgs
One of the following, that corresponds to the transformation you specified above for xformType.
• direct: an empty list (a list with zero elements).
• linear: a list containing two elements: (multiply, add) where multiply is the amount to multiply the original value, and add is the amount to add to the original value.
• range: a list containing these six elements: (modbusMin, modbusMax, pointMin, pointMax, clampMin, clampMax) where:
  • modbusMin is the minimum value for the Modbus value.
  • modbusMax is the maximum value for the Modbus value.
  • pointMin is the minimum value for the point value.
  • pointMax is the maximum value for the point value.
  • clampMin is the minimum clamp value.
  • clampMax is the maximum clamp value.

conversion
A list of (enabled, targetType), where:
• `enabled` is 0 or 1
• `targetType` is a type name, as a string, such as "i4" or "r8"

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command configures a Modbus point. For more information about Modbus data points, please refer to the Data Points section of the Modbus option of the Properties window.
ModbusSlaveAddRange

ModbusSlaveAddRange — configures a range of Modbus points.

Synopsis

(ModbusSlaveAddRange slaveName pointName blockType pointType address allowWrite xformType xformArgs conversion itemCount, nameStart)

Arguments

slaveName
The name of the Modbus slave, as a string.

pointName
A pattern that contains the sequence {0} in at least one place. This sequence will be replaced by an incrementing number for each point in the range. The pattern may optionally include :Dn after the 0, to indicate the number of digits. For example, for sequence number 7, {0} will produce 7 and {0:D3} will produce 007. For example, a point name could be myPoint{0:D3}_Sp.

blockType
A string, one of
- MB_DI for digital input
- MB_DO for digital output
- MB_AI for analog input
- MB_AO for analog output

pointType
The type of point, as a string, consisting of typesize.flags, where:
- type is one of:
  - i for integer
  - r for real (floating point)
  - s for string
  - b for bit
- size is one of:
  - 1 for one bit (b) or one byte (s)
  - 2 for two bytes (i or s)
  - 4 for four bytes (i or r)
  - 8 for eight bytes (i or r)
• *flags* is any combination of
  • – (a minus sign) to indicate a signed integer (i)
  • b to swap bytes within words for 2, 4, or 8 byte sizes
  • w to swap words within dwords for 4 or 8 byte sizes
  • d to swap dwords within quadwords for 8 byte sizes
  • u to indicate UTF-8 for string (s) types of size 1, otherwise ASCII. s2 strings are always UTF-16.
  For example, "i4.-b" would be a signed, 4-bit integer with bytes swapped within words.

**address**
The Modbus address, as a string, consisting of one of the following:
• number - the Modbus address offset, starting from 0 or 1 depending on the slave definition
• number.bit - a single bit within the integer starting at address number
• number.bit-bit - a bit field consisting of all bits between the first and last, inclusive
• number[length] - an array, starting at address number

**allowWrite**
0 for not writable, 1 for writable

**xformType**
The type of transformation to make, one of: *direct*, *linear* or *range*

For more information about transformations, please refer to the Transform section of the Modbus option of the Properties window.

**xformArgs**
One of the following, that corresponds to the transformation you specified above for xformType.
• direct: an empty list (a list with zero elements).
• linear: a list containing two elements: (multiply, add) where multiply is the amount to multiply the original value, and add is the amount to add to the original value.
• range: a list containing these six elements: (modbusMin, modbusMax, pointMin, pointMax, clampMin, clampMax) where:
  • modbusMin is the minimum value for the Modbus value.
  • modbusMax is the maximum value for the Modbus value.
  • pointMin is the minimum value for the point value.
  • pointMax is the maximum value for the point value.
• \textit{clampMin} is the minimum clamp value.
• \textit{clampMax} is the maximum clamp value.

\textit{conversion}

A list of (\textit{enabled}, \textit{targetType}), where:

• \textit{enabled} is 0 or 1
• \textit{targetType} is a type name, as a string, such as "i4" or "r8"

\textit{itemCount}

A number greater than 0 indicating the number of points to define.

\textit{nameStart}

A number indicating the starting number for point name incremental numbering.

\textbf{Returns}

A message indicating success or error. Please refer to \textit{Return Syntax} for details.

\textbf{Description}

This command configures a range of Modbus points. For more information about ranges, please refer to the \textit{Ranges} section of the Modbus option of the Properties window.
**ModbusSlaveDeletePoint**

ModbusSlaveDeletePoint — deletes point connections.

**Synopsis**

```plaintext
(ModbusSlaveDeletePoint slaveName pointNamePattern)
```

**Arguments**

- `slaveName`
  - The name of a Modbus slave connection, as a string.

- `pointNamePattern`
  - An alphanumeric string that matches the name(s) of one or more points.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command removes the connection in the DataHub for one or more points on the Modbus slave.
mult

mult — multiplies the value of a point.

Synopsis

(mult name number secs nano)

Arguments

name
   The name of a point, which must be a number type.

number
   A value to multiply by the value of the point.

secs
   The time in seconds. If this is not specified, the current date and time in seconds is used.

nano
   A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to multiply the value of a point.
OPCActivate

**OPCActivate** — activates or deactivates an OPC group.

**Synopsis**

```plaintext
(OPCActivate label 0|1)
```

**Arguments**

`label`

The name for this connection, as displayed in the **OPC** option of the Properties window. If `label` is `*` then the command affects all currently configured connections.

`0|1`

If `1`, the group is activated; if `0`, it is deactivated.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command activates or deactivates the OPC group in the OPC server representing the connection specified by `label`. This sends a message to the OPC server to activate or deactivate the underlying group. The result of deactivating a group is that subsequent read and write attempts will fail until the group is activated again.
OPCAddItem2

**OPCAddItem2** — adds OPC items to a connection.

**Synopsis**

```python
(OPCAddItem2 label flags propid item (parent...))
```

**Arguments**

- `label`
  The name for this connection, as displayed in the OPC option of the Properties window.

- `flags`
  One or more of the following:

  - `IS_BRANCH` = 0X0001
  - `IS_LEAF` = 0X0002
  - `IS_PROP` = 0X0004
  - `PARENT_IS_LEAF` = 0X0008
  - `IS_EXPLICIT` = 0X0010

  The `IS_EXPLICIT` flag will be added automatically when the DataHub writes the configuration file if it is not present.

- `propid`
  If this point is a property of an OPC leaf item (IS_PROP is true), then the property ID must be entered here. Otherwise, enter 0. This entry is ignored if IS_PROP is not true.

- `item`
  The name of the item on the OPC server, as a string.

- `(parent...)`
  A list of parent DataHub points (OPC branch nodes) that lead to this point, each as a string. If the point is a property (has a non-zero propid), then the last element of the list should be an OPC leaf node. OPC servers can use the "." character in item names, so this hierarchy is not necessarily derivable from the point name.

**Returns**

A message indicating success or error. Please refer to **Return Syntax** for details.
OPCAddItem2

Description

This command adds OPC items to an OPC server connection. It does not take effect until the **OPCApply** command is issued.

Example

```lisp
(OPCAddItem2 "MyOPCServer" 2 0 "Tank3.Level" ("Tank3" "Level"))
```
OPCAEAttach

**OPCAEAttach** — creates an OPC A&E connection.

**Synopsis**

```plaintext
(OPCAEAttach label machine protocol server domain detail_domain flags retry_secs delay_secs)
```

**Arguments**

Many of these parameters correspond to a similarly named entry field or checkbox in the Configure OPC A&E Server dialog options in the OPC A&E option of the Properties window. The names that appear in that dialog are shown here in parentheses.

- **label**
  
  A name used by the Cogent DataHub to identify the connection. (Connection Name)

- **machine**
  
  The name or IP address of the computer running the OPC A&E server you want to connect to. (Computer Name)

- **protocol**
  
  Is not yet documented.

- **server**
  
  The name of the OPC A&E server that you are connecting to. (OPC Server Name)

- **domain**
  
  The name of the DataHub domain in which the data points are received. (Data Domain Name)

- **detail_domain**
  
  Is not yet documented.

- **flags**
  
  Are not yet documented.

- **retry_secs**
  
  The number of milliseconds to wait before retrying a failed connection. (Retry Delay)

- **delay_secs**
  
  The number of milliseconds to wait before retrying a failed connection. (Retry Delay)

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.
Description

This command creates an OPC A&E connection, which corresponds to creating a new connection in the OPC A&E option of the Properties window.
OPCAEDetach

**OPCAEDetach** — deletes an OPC A&E connection.

**Synopsis**

```
(OPCAEDetach label)
```

**Arguments**

`label`

The name used by the Cogent DataHub to identify the connection.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command deletes an OPC A&E connection, like pressing the **Remove** button for a selected OPC A&E connection in the **OPC A&E** option of the Properties window.
OPCAEEnable

OPCAEEnable — enables an OPC A&E connection.

Synopsis

(OPCAEEnable label 0|1)

Arguments

label
The name used by the Cogent DataHub to identify the connection.

0|1
1 enables the connection, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables an OPC A&E connection, like checking a check box in the OPC A&E client list in the OPC A&E option of the Properties window.
OPCAEEnableClient

**OPCAEEnableClient** — enables OPC A&E client behavior.

**Synopsis**

```plaintext
(OPCAEEnableClient 0|1)
```

**Arguments**

0 | 1

1 enables the behavior, 0 disables it.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command enables OPC A&E client behavior, corresponding to the **Act as an OPC A&E Client to these servers** checkbox in the OPC A&E option of the Properties window.
OPCAEEnableServer

OPCAEEnableServer — enables OPC A&E server behavior.

Synopsis

(OPCAEEnableServer 0|1)

Arguments

0|1

1 enables the behavior, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables OPC A&E server behavior, corresponding to the Act as an OPC A&E Server checkbox in the OPC A&E option of the Properties window.
OPCAEFiler

**OPCAEFiler** — selects alarms and events by filtering criteria.

**Synopsis**

```
(OPCAEFiler label filters event_types minseverity maxseverity
 categories areas sources)
```

**Arguments**

Some of these parameters correspond to a similarly named entry field or checkbox in the Configure OPC A&E Server dialog options in the OPC A&E option of the Properties window. The names that appear in that dialog are shown here in parentheses.

- **label**
  The name used by the Cogent DataHub to identify the connection.

- **filters**
  Not yet documented.

- **event_types**
  Three types of filtering options available (not yet documented). (**Filter by event type**)
  - Simple - events not related to an alarm, and cannot be tracked.
  - Tracking - events that originate outside the process being monitored, for example, an operator intervention.
  - Condition - events that indicate that an alarm has been triggered.

- **minseverity**
  An integer specifying the lowest configurable urgency for an alarm. (**Filter by severity - Minimum**)

- **maxseverity**
  An integer specifying the highest configurable urgency for an alarm. (**Filter by severity - Maximum**)

- **categories**
  Not yet documented. (**Filter by category**)

- **areas**
  Not yet documented. (**Filter by area**)

- **sources**
  Not yet documented. (**Filter by source**)

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.
Description

This command selects alarms and events by filtering criteria. It corresponds to using the Filters checkboxes in the OPC A&E option of the Properties window.
OPCAEServerInit

OPCAEServerInit — configures the OPC A&E server.

Synopsis

```
(OPCAEServerInit input_domain detail_domain flags)
```

Arguments

input_domain
The name of the DataHub domain in which the data points are received (corresponds to Use data from this domain in the user interface.)

detail_domain
Is not yet documented.

flags
Have not yet been documented.

• (Accept incomplete acknowledgement information)
• (Send a shutdown to clients when event configuration changes)
• (Automatically refresh conditions when a client connects)

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command configures the OPC A&E server, and corresponds to the Act as an OPC A&E Server option in the OPC A&E option of the Properties window.-->
OPCApply

OPCApply — applies changes to an outgoing connection.

Synopsis

(OPCApply [label])

Arguments

label
The name for this connection, as displayed in the OPC option of the Properties window. If label is * or absent, then the command affects all currently configured connections.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command applies any changes made to an outgoing OPC connection specified by label. The only changes that are not immediately applied are OPCAddItem and OPCRemoveItem. Therefore, you can call OPCAddItem and OPCRemoveItem multiple times and then must issue OPCApply once for the changes to take effect.
**OPCAttach2**

**OPCAttach2** — sets up an OPC connection.

**Synopsis**

```
(OPCAttach2 label machine_name interface_name server_pattern domain
  point_pattern deadband_msec flags pollmsec read_method
  write_method (filters...) [connect_wait ready_wait])
```

**Arguments**

- **label**
  The name for this connection, as displayed in the OPC option of the Properties window.

- **machine_name**
  The name or IP address of the computer running the DataHub.

- **interface_name**
  Is either: "OPC Data Access 2.0" or "OPC Data Access 3.0". Currently only "OPC Data Access 2.0" is supported.

- **server_pattern**
  The name or IP address of the server computer. Wildcard characters are allowed.

- **domain**
  The data domain name.

- **point_pattern**
  A point name filter. Wildcard characters are allowed. Only leaf items that match this pattern will be visible in the DataHub. Normally you should specify * as the pattern.

- **deadband_msec**
  The maximum rate at which the server should send data to the client. In asynchronous advise mode, deadband_msec is transmitted to the server. The server limits the data rate to no more often than the deadband_msec. In polling modes, deadband_msec determines the polling rate.

- **flags**
  Any combination of:
  - 0x00000001 - PROPERTIES - Attempt to load item properties as items.
  - 0x00000002 - ENABLED - This connection is enabled.
  - 0x00000004 - AUTOITEMS - Tells the DataHub to read the entire data set from the server.
  - 0x00000008 - READ_ONLY - Marks the connection as read-only. Data will not be
transmitted from the DataHub to the server.

- 0x00000010 - MANUAL_ITEMS - Tells the DataHub to connect to any manually configured items from the server.
- 0x00000020 - OVERRIDE_TIME - Force the time stamp on incoming data from the server to the local time on the DataHub's computer.

pollmsec
The number of milliseconds between reconnection attempts when trying to connect to a server. This parameter is currently ignored.

read_method
An integer that specifies the method to use to read data from the OPC server. This can be one of:

- 1 - Asynchronous Advise (DA2.0)
- 2 - Synchronous Cache Read (DA2.0)
- 3 - Asynchronous Read (DA2.0)
- 7 - Synchronous Device Read (DA2.0)

write_method
An integer that specifies the method to use to read data from the OPC server. This can be one of:

- 1 - Synchronous Write (DA2.0)
- 2 - Asynchronous Write (DA2.0)

(filters...)
A space-separated list of filters, each one as a string, such as are entered in the Define OPC Server dialog of the OPC option of the Properties window. For example, here is what you would enter if you have:

- zero filters ()
- one filter ("filter1")
- two filters ("filter1","filter2")
- etc.

connect_wait
Optional. The number of milliseconds to pause after the connection is made before continuing with the connection sequence. This gives the server time to start up and construct its data set. If specified, the connect_wait time has to transpire before the DataHub first queries the server's data set.

ready_wait
Optional. The maximum number of milliseconds to wait for the server to report a ready state after the connect_wait has expired. If the server reports that it is ready before the ready_wait time expires, the DataHub will move on immediately to the next step in the connection sequence. If not, then the full ready_wait time will transpire before the DataHub queries the server's data set.
Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you set up an OPC connection. It corresponds roughly to the Define OPC Server dialog box that you get when you click the Add button in the OPC option of the Properties window.
OPCConnect

**OPCConnect** — connects or disconnects from the OPC server.

**Synopsis**

(OPCConnect label 0|1)

**Arguments**

*label*

The name for this connection, as displayed in the OPC option of the Properties window. If *label* is * then the command affects all currently configured connections.

*0|1*

If 1, connect; if 0, disconnect.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command causes the outgoing OPC connection specified by *label* to connect to or disconnect from the OPC server. If the connection is disconnected, it will reconnect again according to its connection polling cycle. If you wish to disconnect such that the connection will not automatically be retried, use **OPCEnable**.
OPCDetach

OPCDetach — removes an outgoing OPC connection.

Synopsis

```
(OPCDetach label)
```

Arguments

`label`

The Connection name as set by `OPCAttach` or displayed in the OPC option of the Properties window.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command detaches and deletes the outgoing OPC connection specified by `label`. 
OPCEnable — enables or disables outgoing OPC connections.

Synopsis

(OPCEnable label 0|1)

Arguments

label
The name for this connection, as displayed in the OPC option of the Properties window. If label is * then the command affects all currently configured connections.

0|1
If 1, the connection is enabled; if 0, it is disabled.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

If the argument is 1, the outgoing OPC connection specified by the label is enabled. This is equivalent to checking the check-box for the specified OPC connection in the OPC option of the Properties window. If the argument is 0, the specified connection is disabled.
OPCEnableClient

**OPCEnableClient** — enables or disables all OPC clients.

**Synopsis**

```
(OPCEnableClient 0|1)
```

**Arguments**

0|1

If 1, all OPC client connections that are individually enabled will be immediately started. If 0, all OPC client connections (outgoing) are terminated.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.
**OPCEnableServer**

**OPCEnableServer** — enables or disables DataHub OPC server behavior.

**Synopsis**

```
(OPCEnableServer 0|1)
```

**Arguments**

0|1

If 1, the OPC DataHub will be added to the list of available OPC servers on this computer and future incoming OPC connections will be accepted. If 0, the OPC DataHub will be removed from the list of available OPC servers on this computer and all future incoming OPC connections will be denied when the client attempts to create an OPC group.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.
**OPCMinimumSecurity**

**OPCMinimumSecurity** — overrides DCOM security settings.

**Synopsis**

```
(OPCMinimumSecurity 0|1)
```

**Arguments**

0|1

If 1, the OPC DataHub will attempt to override DCOM security settings. If 0, it will not.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command determines whether the OPC DataHub attempts to override the current system COM security settings when it starts. If you change this setting, you must re-start the OPC DataHub for the change to be effective. An argument of 1 is equivalent to checking the check-box **Attempt to override application DCOM setting with minimum security settings** in the OPC option of the Properties window.
OPCModify

**OPCModify** — modifies an existing OPC connection.

**Synopsis**

```plaintext
(OPCModify label machine_name interface_name server_pattern domain
point_pattern deadband_msec flags pollmsec read_method
write_method (filters...))
```

**Arguments**

**label**

The name for this connection, as displayed in the **OPC** option of the Properties window.

**machine_name**

The name or IP address of the computer running the DataHub.

**interface_name**

Is currently ignored.

**server_pattern**

The name or IP address of the server computer. Wildcard characters are allowed.

**domain**

The data domain name.

**point_pattern**

A point name filter. Wildcard characters are allowed. Only leaf items that match this pattern will be visible in the DataHub. Normally you should specify * as the pattern.

**deadband_msec**

The maximum rate at which the server should send data to the client. In asynchronous advise mode, `deadband_msec` is transmitted to the server. The server limits the data rate to no more often than the `deadband_msec`. In polling modes, `deadband_msec` determines the polling rate.

**flags**

Any combination of:

- 0x00000001 - PROPERTIES - Attempt to load item properties as items.
- 0x00000002 - ENABLED - This connection is enabled.
- 0x00000004 - AUTOITEMS - Tells the OPC DataHub to read the entire data set from the server.
- 0x00000008 - READ_ONLY - Marks the connection as read-only. Data will not be transmitted from the OPC DataHub to the server.
• 0x00000010 - MANUAL_ITEMS - Tells the OPC DataHub to connect to any manually configured items from the server.
• 0x00000020 - OVERRIDE_TIME - Force the time stamp on incoming data from the server to the local time on the OPC DataHub's computer.

`pollmsec`
The number of milliseconds between reconnection attempts when trying to connect to a server.

`read_method`
The method to use to read data from the OPC server. This can be one of:
• 1 - Asynchronous Advise (DA2.0)
• 2 - Synchronous Cache Read (DA2.0)
• 3 - Asynchronous Read (DA2.0)
• 7 - Synchronous Device Read (DA2.0)

`write_method`
The method to use to read data from the OPC server. This can be one of:
• 1 - Synchronous Write (DA2.0)
• 2 - Asynchronous Write (DA2.0)

`filters...`
A space-separated list of filters, each one as a string, such as are entered in the Define OPC Server dialog of the OPC option of the Properties window. For example, here is what you would enter if you have:
• zero filters ()
• one filter ("filter1")
• two filters ("filter1" "filter2")
• etc.

**Returns**
A message indicating success or error. Please refer to Return Syntax for details.

**Description**
This command is identical to **OPCAttach2**, except that it applies its settings to an existing connection instead of creating a new connection.
OPCQueryConnection

**OPCQueryConnection** — provides information related to an OPC server connection.

**Synopsis**

```
(OPCQueryConnection label)
```

**Arguments**

`label`

A name used by the Cogent DataHub to identify the connection, as listed in the **Connection** column of the **OPC DA** option of the Properties window.

**Returns**

A message indicating success or error. Please refer to [Return Syntax](#) for details.

**Description**

This command provides information related to an OPC server connection. A detailed description of the values provided has not yet been documented.

**Example**

This is an example of the command being called from the Script Log, using the `datahub_command` function. A few carriage returns have been added to this example to make the output easier to read.

```bash
--> pretty_princ(datahub_command ("(OPCQueryConnection OPC004)", 1));
(OPCQueryConnection "OPC004" "localhost" "OPC Data Access 2.0"
 "Toolbox OPC Power Server 5.14" "TOP5" "+" 500 0x700312
 5000 1 2 () 1000 5000 "Running" 0x3f9f8f0)
```
**OPCQueryConnections**

**OPCQueryConnections** — lists OPC server connections.

**Synopsis**

```plaintext
(OPCQueryConnections [pattern])
```

**Arguments**

`pattern`

A pattern that matches names of OPC connections, such as OPC00*.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command lists OPC server connections.

**Example**

This is an example of the command being called from the Script Log, using the `datahub_command` function.

```plaintext
--> pretty_princ(datahub_command ("(OPCQueryConnections OPC00*)", 1));
(.OPCQueryConnection "OPC000" "OPC001" "OPC004")
```
OPCQueryPoint

**OPCQueryPoint** — provides data about an OPC point.

**Synopsis**

```
(OPCQueryPoint label pointname)
```

**Arguments**

*label*  
A name used by the Cogent DataHub to identify the connection, as listed in the **Connection** column of the **OPC DA** option of the Properties window.

*pointname*  
The name of a DataHub point only (no domain name) within the domain that has been configured for the OPC connection.

**Returns**

A message indicating success or error. Please refer to **Return Syntax** for details.

**Description**

This command provides data about a point within a configured OPC connection. Details regarding the nature of the data have not been documented.

**Example**

This is an example of the command being called from the Script Log, using the **datahub_command** function. A few carriage returns have been added to this example to make the output easier to read.

```bash
-- pretty_princ(datahub_command
    "((OPCQueryPoint OPC004 Channel_0.Ramp.Ramp1)", 1));
(OPCQueryPoint OPC004 ("OPC004" "Channel_0.Ramp.Ramp1" 18 0
"Channel_0.Ramp.Ramp1" ("Channel_0" "Ramp" "Ramp1")))
```
**OPCQueryPointPattern**

*OPCQueryPointPattern* — provides data about multiple OPC points.

**Synopsis**

{(OPCQueryPointPattern label pattern)}

**Arguments**

*label*

A name used by the Cogent DataHub to identify the connection, as listed in the **Connection** column of the **OPC DA** option of the Properties window.

*pattern*

A pattern that matches names of OPC connections, such as **OPC00**.*

**Returns**

A message indicating success or error. Please refer to **Return Syntax** for details.

**Description**

This command provides data about multiple OPC points.

**Example**

This is an example of the command being called from the Script Log, using the *datahub_command* function. A few carriage returns have been added to this example to make the output easier to read.

```bash
--> pretty_princ(datahub_command
    "((OPCQueryPointPattern OPC004 *Ramp*)", 1));
"(OPCQueryPointPattern OPC004
  ("OPC004" "Channel_0.Ramp.Ramp1" 18 0 "Channel_0.Ramp.Ramp1"
   ("Channel_0" "Ramp" "Ramp1"))
  ("OPC004" "Channel_0.Ramp.Ramp2" 18 0 "Channel_0.Ramp.Ramp2"
   ("Channel_0" "Ramp" "Ramp2"))
  ("OPC004" "Channel_0.Ramp.Ramp3" 18 0 "Channel_0.Ramp.Ramp3"
   ("Channel_0" "Ramp" "Ramp3"))
)"
```
OPCQueryPoints

OPCQueryPoints — provides a list of explicitly configured OPC tags in a connection.

Synopsis

(OPCQueryPoints label [pattern])

Arguments

label
A name used by the Cogent DataHub to identify the connection, as listed in the Connection column of the OPC DA option of the Properties window.

pattern
A pattern that matches names of OPC connections, such as OPC00*.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command provides a list of OPC tags that have been explicitly (manually) configured for a connection. Details regarding the nature of the data have not been documented.

Example

This is an example of the command being called from the Script Log, using the datahub_command function. A few carriage returns have been added to this example to make the output easier to read.

```ruby
--> pretty_princ(datahub_command
   ((OPCQueryPoints OPC004", 1));
(OPCQueryPoints OPC004 "Channel_0.Ramp.Ramp1"
   "Channel_0.Ramp.Ramp2"
   "Channel_0.Ramp.Ramp3"
   "_System")
```
OPCRefresh

OPCRefresh — sends a Refresh2 command to the OPC server.

Synopsis

(OPCRefresh [label])

Arguments

label
The name for this connection, as displayed in the OPC option of the Properties window. If label is * or absent, then the command affects all currently configured connections.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command causes the outgoing OPC connection specified by label to send an OPC Refresh2 command to the server. This instructs the OPC server to retransmit the current values for all items to which this connection is subscribed.
OPCReload

**OPCReload** — reloads the data set from an OPC server.

**Synopsis**

```plaintext
(OPCReload [label [reconnect]])
```

**Arguments**

- `label`
  The name for this connection, as displayed in the OPC option of the Properties window. If `label` is * or absent, then the command affects all currently configured connections.

- `reconnect`
  Either 1 or no argument causes a disconnect and reconnect during the reload cycle. A 0 prevents a disconnect from the OPC server during reload.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command causes the outgoing OPC connection specified by `label` to reload its data set from the OPC server. If the `reconnect` argument is absent or is 1, the connection will be disconnected and then reconnected during the reload cycle. If `reconnect` is 0, then the data set will be reloaded without disconnecting from the server.
OPCRemovItem

**OPCRemovItem** — removes an item based on its OPC DataHub point name.

**Synopsis**

```text
(OPCRemovItem label point_name)
```

**Arguments**

- **label**
  The name for this connection, as displayed in the OPC option of the Properties window.

- **point_name**
  The name of an OPC DataHub point, as a string, without the data domain name.

**Returns**

A message indicating success or error. Please refer to *Return Syntax* for details.

**Description**

This command removes an OPC item based on its OPC DataHub point name. OPC items are mapped in a one-to-many relationship with OPC DataHub points. The command removes only the item mapping associated with the given point name. The point name must not be qualified with the data domain name. The `label` is the label supplied to `OPCAttach` or `OPCAttach2`.

This command does not take effect until the `OPCApply` command is issued.
**private_attribute**

private_attribute — creates a private attribute.

**Synopsis**

```plaintext
(private_attribute assemblyname attrname type rw dflt_value [dflt_conf])
```

**Arguments**

- **domain**
  The domain to which this property applies.

- **assemblyname**
  The name of the assembly to which this attribute applies.

- **attrname**
  The name of the attribute.

- **type**
  A type for the private attribute.

- **rw**
  One of:
  - *r* for read-only.
  - *w* for write-only.
  - *rw* for read-write.

- **dflt_value**
  A default value.

- **dflt_conf**
  A default confidence level. If nothing is entered, the system assumes 0.

**Returns**

A message indicating success or error. Please refer to *Return Syntax* for details.

**Description**

This command creates a private attribute. For information on the difference between an attribute and a private attribute, please refer to the section called “Attributes and Types”.
**property**

`property` — creates a property for an assembly.

**Synopsis**

```
(property domain attrname propid propname type rw dflt_value
[dflt_conf])
```

**Arguments**

- **domain**
  The domain to which this property applies.

- **attrname**
  The name of the attribute to which this property applies.

- **propid**
  An ID number, or AUTO to have the DataHub assign an ID automatically.

- **propname**
  A name for the property.

- **type**
  A type for the property.

- **rw**
  One of:
  - `r` for read-only.
  - `w` for write-only.
  - `rw` for read-write.

- **dflt_value**
  A default value.

- **dflt_conf**
  A default confidence level. If nothing is entered, the system assumes 0.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command creates a property. For more information and an example, please refer to the section called “Assemblies, Subassemblies, Attributes, and Properties”.
quality

**quality** — assigns a quality to a point.

**Synopsis**

```scheme
(quality name new_quality)
```

**Arguments**

- `name`:
  The name of a point, as a string.

- `new_quality`:
  The quality to be assigned to the point, as a number. Quality numbers are:

  ```
  OPC_QUALITY_BAD = 0
  OPC_QUALITY_UNCERTAIN = 0x40
  OPC_QUALITY_GOOD = 0xc0
  OPC_QUALITY_CONFIG_ERROR = 0x4
  OPC_QUALITY_NOT_CONNECTED = 0x8
  OPC_QUALITY_DEVICE_FAILURE = 0xc
  OPC_QUALITY_SENSOR_FAILURE = 0x10
  OPC_QUALITY_LAST_KNOWN = 0x14
  OPC_QUALITY_COMM_FAILURE = 0x18
  OPC_QUALITY_OUT_OF_SERVICE = 0x1c
  OPC_QUALITY_WAITING_FOR_INITIAL_DATA = 0x20
  OPC_QUALITY_LAST_USABLE = 0x44
  OPC_QUALITY_SENSOR_CAL = 0x50
  OPC_QUALITY_EGU_EXCEEDED = 0x54
  OPC_QUALITY_SUB_NORMAL = 0x58
  OPC_QUALITY_LOCAL_OVERRIDE = 0xd8
  ```

**Returns**

A message indicating success or error. Please refer to **Return Syntax** for details.

**Description**

This command assigns a quality to a point. Typical qualities include **Good** and **Bad**.
**read**

*read* — reads a complete point definition.

**Synopsis**

```plaintext
(read name)
```

**Arguments**

`name`

The name of a point.

**Returns**

The complete point definition in a message, with this syntax:

```plaintext
(point name type value [conf security locked seconds nanoseconds flags quality])
```

where:

`name`

The name of the point.

`type`

The data type of the point, one of integer, floating point, or character string.

`value`

The value of the point.

`conf`

The confidence level of the point, 0 - 100 percent, unused by most applications.

`security`

The security level of the point, 0 to 32768, where higher numbers represent higher security.

`locked`

0 for locked, or 1 for unlocked.

`seconds`

The operating system time in seconds when the point was read.

`nanoseconds`

The number of nanoseconds after `seconds` when the point was read.

`flags`

User-defined flags.
quality
A constant representing a quality of the connection, assigned by the DataHub for this point, such as Good, Bad, Last known, Local override, etc. The possible values are those supported by OPC in Microsoft Windows.

Description
This command reads the current value of a point, along with all the other information it contains. See also cread.
Report

Report — requests notification of changes to a data point.

Synopsis

(report name)

Arguments

name

The name of a data point.

Returns

A message with the complete definition of the point.

Description

This command requests the DataHub to report changes (also called exceptions) to the value or any other information about a point, as soon as any change takes place. See also creport.
**report_domain**

**Synopsis**

```
(report_domain domain flags)
```

**Arguments**

*domain*

The desired domain.

*flags*

Any bitwise-or combination of:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DH_REG_ALL</td>
<td>Register all points on this domain.</td>
</tr>
<tr>
<td>DH_REG_FUTURE</td>
<td>Register any new points created on this domain subsequent to this call.</td>
</tr>
<tr>
<td>DH_REG_QUALIFY</td>
<td>Tell the DataHub to emit all point names with the domain prepended, as in <em>domain:point_name</em>.</td>
</tr>
<tr>
<td>DH_REG_ONCEONLY</td>
<td>Tell the DataHub to send each point value exactly once.</td>
</tr>
<tr>
<td>DH_REG_MODEL</td>
<td>Tell the DataHub to send the data model as well as the point values.</td>
</tr>
</tbody>
</table>

**Returns**

One or more messages, depending on the *flag(s)* chosen. Each message contains the complete definition of the point.

**Description**

This command lets TCP client connections decide on a per-domain basis whether to informed of the data model for the domain, and whether to get future updates, fully-qualified domain names, or new points.
**report_errors**

- **report_errors** — controls the reporting of errors.

**Synopsis**

```
(report_errors 0|1 [error_point])
```

**Arguments**

- **0|1**
  - If 0 (the default), errors in transmitting an exception will not be reported to the client.
  - If 1, an error string will be generated.

- **error_point**
  - (Optional) The name of a point to contain the error string. If no name is given here, the error string will be written as the value of the point in which the error occurred.

**Returns**

A message indicating success or error. Please refer to [Return Syntax](#) for details.

**Description**

This command allows a client to specify how a failure to transmit a point change should be reported, either by modifying the point data and trying again, or by emitting an "error" point with the message in its data. The default is no reporting at all. In any case, if the attempt to emit the exception with the error information also fails, no further action is attempted by the DataHub.
request_initial_data

request_initial_data — gets current data when client connection is made.

Synopsis

(request_initial_data yes|no|0|1)

Arguments

yes|no|0|1

Choose yes or 1 to request the data, or no or 0 to not make the request.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command causes the DataHub to send the current value of all points when the client connects.
save_config

**Synopsis**

{save_config} — forces the DataHub to save its configuration.

**Arguments**

none

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command allows an external client to force the DataHub to save its configuration.
secure

secure — adjusts the security level of a point.

Synopsis

(security name my_sec new_sec)

Arguments

name
The name of a DataHub point, as a string.

my_sec
The user’s security level.

new_sec
The new security level to be assigned to the point.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command adjusts the security level of a DataHub point. If a point has a non-zero security level then any attempt to change the point value will fail if the writer claims to have a security level lower than the point’s security level. The term "security level" is something of a misnomer because it is up to the writer to state what security level it has, and the writer could claim to have any security level. There is no validation of the claim by the DataHub. Consequently, this security level is co-operative. It acts to stop errors among trusted programs, but does not act to limit access by untrusted programs.

The default security level for a point is 0.
set

set — sets the value of a point.

Synopsis

(set name value [confidence])

Arguments

name

The name of the point. This is a string.

value

A string representation of the value for the point. The point value will be interpreted as integer, float or string based on the contents of the value string. This function tries each type in order, and uses the first type for which the value parameter is a valid representation. Double quotes around the value parameter are ignored. For example:

• 123 is an integer.
• 123.4 is a float.
• 1.234e2 is a float.
• "123" is an integer.
• 123abc is a string.

confidence

A confidence factor in the range of 0 to 100 (optional). This is not used by the DataHub, so is available to programs that produce graduated confidence, such as expert systems. If this value is not specified, it is set to 100.

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (\) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with \n, \r, \f, \t respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets the value of a point. When this value is set, the following attributes of the point are set as follows:

• seconds and nanoseconds are set to the current time on the machine running the DataHub.
• locked, sec, and quality are all maintained at their previous values for this point.
• flags is set to 0.

Please refer to the write command for more information about these parameters. See also cset, force, and cforce.
set_access

set_access — is new, not yet documented.

Synopsis

(set_access pointname rw)

Description

This command has not yet been documented.
set_authoritative

**set_authoritative** — sets the type of a point.

**Synopsis**

```
(set_authoritative domain flag)
```

**Arguments**

- **domain**
  - The name of the data domain.
- **flag**
  - A value of 1 indicates that this application is authoritative. A value of 0 indicates that the DataHub should be authoritative.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command tells the DataHub that this application should be considered the authoritative source of data for this data domain. The DataHub will keep track of which data points this application has written to the DataHub, and will automatically mark those points as being Not Connected quality when the application disconnects. This provides a mechanism for the DataHub to provide quality information even if the source application exits unexpectedly.

There should only ever be one authoritative source of data for any data point, though the DataHub will not enforce this. If more than one application claims to be authoritative for a data point, that point will be given a Not Connected quality even if another authoritative application is still connected.

The command `(set_authoritative domain 0)` does NOT mean "turn off authoritative status". It informs the DataHub that the application is explicitly non-authoritative, and by extension that the DataHub should behave as if it is authoritative. Normally this is only meaningful between DataHub instances constructing a tunnel. An application should not send `(set_authoritative domain 0)` to the DataHub. If an application does not want to be authoritative for a data domain, it simply should not send a **set_authoritative** command.
**set_canonical**

**set_canonical** — sets the type of a point.

**Synopsis**

```
(set_canonical pointname canonical_type [force])
```

**Arguments**

- **pointname**
  The full name of the point, with domain.

- **canonical_type**
  Either a number with a legal numeric VT_TYPE value, or one of: I1, I2, I4, UI1, UI2, UI4, CY, DATE, BOOL, BSTR, R4, R8, EMPTY, I1 ARRAY, I2 ARRAY, I4 ARRAY, UI1 ARRAY, UI2 ARRAY, UI4 ARRAY, CY ARRAY, DATE ARRAY, BOOL ARRAY, BSTR ARRAY, R4 ARRAY, R8 ARRAY.

- **force**
  A value of 1 forces a change in canonical type. A value of 0, or the omission of this optional parameter, will allow the canonical type to change only if it is currently EMPTY.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command sets the canonical type of a point. Normally a point’s canonical type is EMPTY, meaning that it will maintain the data type of whatever data is written to it. If the canonical type is other than EMPTY, then any data written to the point will be converted to that type before it is stored.

When a point has a non-empty canonical type it is possible that the conversion could fail, in which case the point value is not changed.
show_data

show_data — displays the Data Browser.

Synopsis

(show_data 0|1)

Arguments

0|1

Use 1 to show the Data Browser, or 0 to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command shows or hides the Data Browser.
show_debug_messages

show_debug_messages — show or hide debugging messages in the Data Browser.

Synopsis

(show_debug_messages 0|1)

Arguments

0|1

Use 1 to show debugging messages in the Data Browser, or 0 to not show them.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you toggle the functionality of the Debug button in the Event Log on or off programmatically. The Debug option is very verbose, and could put a high demand on system resources.
**show_event_log**

`show_event_log` — displays the Event Log.

**Synopsis**

```
(show_event_log 0|1)
```

**Arguments**

- **0|1**
  
  Use 1 to show the Event Log, or 0 to hide it.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command shows or hides the Event Log.
show_icon

— displays the system tray icon.

Synopsis

(show_icon 0|1)

Arguments

0|1

Use 1 to display the system tray icon, or 0 to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you show or hide the system tray icon (🔗).
show_properties

show_properties — displays the Properties window.

Synopsis

(show_properties 0|1)

Arguments

0|1

Use 1 to display the Properties window, or 0 to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command shows or hides the Properties window.
**show_script_log**

*show_script_log* — displays the Script Log.

**Synopsis**

```plaintext
(show_script_log 0|1)
```

**Arguments**

0|1

Use 1 to display the Script Log, or 0 to hide it.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command shows or hides the Script Log.
subassembly

subassembly — creates a subassembly.

Synopsis

(subassembly domain assemblyname subassemblyname instancename)

Arguments

domain
   The name of the domain in which this subassembly will be created.

assemblyname
   The name of the parent assembly.

subassemblyname
   The name for this subassembly.

instancename
   The instance name for this instance of the subassembly.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a subassembly level of a data organization. Unlike assemblies, each subassembly is instantiated when it is created, and therefore needs an instance name. For more information about assemblies and subassemblies with an example, please refer to the section called “Assemblies, Subassemblies, Attributes, and Properties”.

success

success — sets up a success message.

Synopsis

```
(success command resultstring)
```

Arguments

- **command**: The name of a command for which this success string will be used.
- **resultstring**: A message string to be send to the issuer of a command when the command executes successfully. Also see Return Syntax.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to create messages that are sent from the DataHub to participating programs whenever a given `command` is successfully executed.
**tcp_service**

* tcp_service — sets a TCP service name or port number for incoming slave connections.

**Synopsis**

```
(tcp_service service|port)
```

**Arguments**

*service|port*

The TCP service name or port number for incoming slave connections.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command informs a DataHub acting as a master which TCP service name or port number it should listen on for incoming slave connections.
timeout

timeout — suspends data flow.

Synopsis

(timeout ms)

Arguments

ms
The number of milliseconds to pause.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command causes the DataHub to stop sending data to the client for ms milliseconds.
transmit_insignificant

**Synopsis**

```plaintext
(transmit_insignificant 0|1)
```

**Arguments**

0|1

Use 1 to transmit insignificant changes, or 0 to not transmit them.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command permits transmission of insignificant changes. A change is considered insignificant if a point change differs from the existing value only by its timestamp. A change is considered significant if any attribute other than timestamp changes. That would include lock, security, confidence, quality, value. This command does not put any kind of deadband on the value.
TunnelDelete

**Synopsis**

```
(TunnelDelete host_pattern [remote_domain_pattern])
```

**Arguments**

- **host_pattern**
  - The name of the host (the remote machine), or a pattern that matches several host names.

- **remote_domain_pattern**
  - The domain on the remote host, or a pattern that matches several domains. If not specified, all domains on that host will be selected.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command deletes all tunnel slave connections whose host name matches `host_pattern` and whose remote domain name matches `remote_domain_pattern`. This corresponds to the **Remove** button in the Tunnel/Mirror option of the Properties window.
TunnelEnable

**TunnelEnable** — enables specified tunnel/mirror Slave connections.

**Synopsis**

```
(TunnelEnable 0|1 host_pattern [remote_domain_pattern])
```

**Arguments**

0|1
1 enables the capability, 0 disables it.

*host_pattern*

The name of the host (the remote machine), or a pattern that matches several host names.

*remote_domain_pattern*

The domain on the remote host, or a pattern that matches several domains.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command enables or disables tunnelling to the host(s) and domain(s) that match a1 pattern. This corresponds to checking the check boxes on configured slave connections in the Tunnel/Mirror option of the Properties window.
TunnelEnablePlain

TunnelEnablePlain — enables plain-text tunnel/mirror Master connections.

Synopsis

(TunnelEnablePlain 0|1)

Arguments

0|1

1 enables the capability, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables plain-text tunnel/mirror Master connections, and corresponds to the Accept plain-text connections on service/port checkbox in the Tunnel/Mirror option of the Properties window.
TunnelEnableSlave

TunnelEnableSlave — enables all tunnel/mirror Slave connections.

Synopsis

(TunnelEnableSlave 0|1)

Arguments

0|1

1 enables the capability, 0 disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables all tunnel/mirror Slave connections, and corresponds to the Act as a tunnel/mirror slave to these masters checkbox in the Tunnel/Mirror option of the Properties window.
**TunnelEnableSSL**

TunnelEnableSSL — enables SSL tunnel/mirror Master connections.

**Synopsis**

```
(TunnelEnableSSL 0|1)
```

**Arguments**

0|1

1 enables the capability, 0 disables it.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command enables SSL tunnel/mirror Master connections, and corresponds to the Accept secure connections on service/port checkbox in the Tunnel/Mirror option of the Properties window.
TunnelPlainPort

TunnelPlainPort — specifies the port for incoming plain-text tunnels.

Synopsis

```
(TunnelPlainPort port)
```

Arguments

*port*
A TCP port number

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command specifies the port for incoming plain-text tunnels, and corresponds to the entry field for *Accept plain-text connections on service/port* checkbox in the Tunnel/Mirror option of the Properties window.
**TunnelSaveConfig**

*TunnelSaveConfig* — saves the tunnel configuration.

**Synopsis**

```plaintext
(TunnelSaveConfig)
```

**Arguments**

None

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command causes the DataHub Tunnel/Mirror plugin to immediately save its current configuration to its configuration file. This will not capture any changes that have been manually configured in the Properties window. Manual changes must be saved by pressing the **Apply** button in that window.
**TunnelSlaveStatus**

*TunnelSlaveStatus* — provides status information on tunnel/mirror Slave connections.

**Synopsis**

```lisp
(TunnelSlaveStatus host_pattern)
```

**Description**

This command provides status information on tunnel/mirror Slave connections. Details regarding the nature of the status data have not been documented.

**Example**

This is an example of the command being called from the Script Log, using the `datahub_command` function. A few carriage returns have been added to this example to make the output easier to read.

```lisp
--> pretty_princ(datahub_command
    ("(TunnelSlaveStatus developers.cogentrts.com") 1));
(TunnelSlaveStatus ("slave" "developers.cogentrts.com" "4502" ""
  "4502" "test" "test" "" 0x22300000 5000 1000 5000 1 "Running"))
```

```
TunnelSSLCert

TunnelSSLCert — specifies the certificate for incoming SSL-enabled tunnels.

Synopsis

(TunnelSSLCert filename)

Arguments

filename

The directory and filename of the SSL certificate.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command specifies the security certificate for incoming SSL-enabled tunnels. This corresponds to the entry field for the SSL Certificate in the Tunnel/Mirror option of the Properties window.
TunnelSSLPort

TunnelSSLPort — specifies the port for incoming SSL-enabled tunnels.

Synopsis

(TunnelSSLPort port)

Arguments

port
A TCP port number

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command specifies the port for incoming SSL-enabled tunnels. This corresponds to the entry field for the Accept Secure connections on service/port checkbox in the Tunnel/Mirror option of the Properties window.
type

  type — creates a type.

Synopsis

    (type domain attrname [superattrname])

Arguments

  domain
  The domain in which this type applies.

  attrname
  The name of this type, which is the same as the name of the attribute.

  superattrname
  An attribute from which this type is derived.

Returns

  A message indicating success or error. Please refer to Return Syntax for details.

Description

  This command creates a type. For more information and an example, please refer to the
  section called “Attributes and Types”. 
unload_plugin

unload_plugin — unloads a plugin. (experimental)

Synopsis

```
(unload_plugin plugin_name)
```

Arguments

`plugin_name`

The name of a plugin.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the DataHub to unload a plugin.

The commands related to plugins are currently experimental.
unreport

**unreport** — allows a client to stop receiving data value changes to a point.

**Synopsis**

```
(unreport name)
```

**Arguments**

- **name**
  The name of the point to stop receiving values for.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command allows a client to stop future changes from being sent on a specified point.
unreport_domain

unreport_domain — allows a client to stop receiving data value changes in a whole domain.

Synopsis

```
(unreport_domain domain)
```

Arguments

domain

The name of a DataHub domain.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

Typically this command is sent by a client to the DataHub, allowing the client to stop receiving data value changes in the specified domain.
version

	version — returns the current version number.

Synopsis

(version)

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command gives the current version number of the DataHub.
**write**

*write* — writes information to a point.

**Synopsis**

```
(write name type value conf sec locked seconds nanoseconds
[flags quality])
```

**Arguments**

**name**

The name of the point. This is a string.

**type**

The type of point. One of

- 0 = string
- 1 = float (8-byte double)
- 2 = integer (32-bit integer)

**value**

A string representation of the value for the point. It will be interpreted into the type specified by the *type* parameter.

**conf**

A confidence factor in the range of 0 to 100. This is not used by the DataHub, so is available to programs that produce graduated confidence, such as expert systems.

**sec**

A security level for this point. This is rarely used. If a point's security level is set to a non-zero value then attempts to write to that point must claim a security level equal to or greater than the security level of the point. This uses a "good citizen" model - the writer can claim any security it wants, and is assumed to be honest - so there is no strong security here. It is intended for systems that want to avoid accidental changes to values. Security level can be from 0 to 32767.

**locked**

An indication that the point is locked, and cannot be changed. Can be 0 or 1. Attempts to write to a locked point will fail.

**seconds**

The UNIX epoch - seconds since Jan. 1, 1970, as produced by the `time()` function.

**nanoseconds**

The number of nanoseconds inside this second. Cannot exceed 1,000,000,000.

**flags**

User level code should always send a 0 for this value.
quality
A quality indicator consistent with the OPC DA specification. This is not a bit field. It can be one of:

- PT_QUALITY_BAD 0
- PT_QUALITY_UNCERTAIN 0x40
- PT_QUALITY_GOOD 0xc0
- PT_QUALITY_CONFIG_ERROR 0x04
- PT_QUALITY_NOT_CONNECTED 0x08
- PT_QUALITYDEVICE_FAILED 0x0c
- PT_QUALITY_SENSOR_FAILURE 0x10
- PT_QUALITY_LAST_KNOWN 0x14
- PT_QUALITY_COMM_FAILURE 0x18
- PT_QUALITY_OUT_OF_SERVICE 0x1c
- PT_QUALITY_WAITING_FOR_INITIAL_DATA 0x20
- PT_QUALITY_LAST_USABLE 0x44
- PT_QUALITY_SENSOR_CAL 0x50
- PT_QUALITY_EGU_EXCEEDED 0x54
- PT_QUALITY_SUB_NORMAL 0x58
- PT_QUALITY_LOCAL_OVERRIDE 0xd8

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (\) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with \n, \r, \f, \t respectively. Strings must not contain newline characters.

Returns
A message indicating success or error. Please refer to Return Syntax for details.

Description
This command lets you manually write information to a point. See also cwrite.
Obsolete and Unused Commands

This reference contains commands that are deprecated, obsolete, and no longer used, as well as commands that are for internal use only. These commands should rarely if ever be used.
asyncsocket

- **asyncsocket** — sets up asynchronous communication on a socket.

**Synopsis**

```bash
(asyncsocket socket)
```

**Arguments**

socket

- The socket address, as a string.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command tells the DataHub to communicate asynchronously on the specified socket.
authgroup — is deprecated.

Synopsis

```
(authgroup name bits max_concurrent_logins max_logins expiry)
```
authuser

authuser — is deprecated.

Synopsis

(authuser name group hash bits max_concurrent_logins max_logins login_count expiry)
bandwidth_reduce

bandwidth_reduce — is for internal use only.

Synopsis
deleted

— checks if a point has been deleted.

Synopsis

(deleted pointname [domain])

Arguments

pointname
A string containing the name of the point.

domain
The domain of the point. If not specified, the default domain is used.

Returns

A message indicating whether the point has been deleted.

Description

This command checks if the given point has been deleted.

We do not recommend deleting points, as it could cause unexpected behavior for other users of the point.
drop_license

*drop_license* — is for internal use only.

Synopsis
echo

echo — is for internal use.

Synopsis

```
{echo name type value [conf security locked seconds nanoseconds flags quality]}
```

Description

This command is only used between two DataHubs. A non-DataHub client should never issue an `echo` command.
enable_connect_server

enable_connect_server — is deprecated.

Synopsis

(enable_connect_server 0|1)
EnableDDEServer

EnableDDEServer — is for internal use only.

Synopsis

(EnableDDEServer 0 | 1)

Description

This command is for internal use only. To enable the DDE server, please refer to enable_dde_server.
exception_buffer

exception_buffer — is deprecated.

Synopsis

```
(exception_buffer bytes)
```
failed_license

failed_license — is for internal use only.

Synopsis
flush

flush — flushes output to a terminal (Linux).

Synopsis

```
(flush)
```

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is not used at all in Windows, and even in Linux and QNX there is really not much need for it. The command sends a flush to the terminal, so that if the DataHub is printing to a terminal and has not completely flushed its output, this will cause all pending printed characters to appear.
flush_log

flush_log — is deprecated.

Synopsis

```bash
{flush_log}
```
**master_host**

`master_host` — is deprecated in favor of `mirror_master`.

**Synopsis**

```
(master_host name|IP)
```
master_service

master_service — is deprecated in favor of mirror_master.

Synopsis

(master_service service|port)
on_change

— is for internal use only.

Synopsis
**OPCAddItem**

*OPCAddItem* — is deprecated in favor of *OPCAddItem2*.

**Synopsis**

```
(OPCAddItem label flags propid item (parent...))
```

**Arguments**

*label*

The name for this connection, as displayed in the *OPC* option of the Properties window.

*flags*

one or more of the following:

- `IS_BRANCH` = 0X0001
- `IS_LEAF` = 0X0002
- `IS_PROP` = 0X0004
- `PARENT_IS_LEAF` = 0X0008

*propid*

If this point is a property of an OPC leaf item (`IS_PROP` is true), then the property ID must be entered here. Otherwise, enter 0. This entry is ignored if `IS_PROP` is not true.

*item*

The name of the item on the OPC server, as a string.

*(parent...)*

A list of parent DataHub points (OPC branch nodes) that lead to this point, each as a string. If the point is a property (has a non-zero `propid`), then the last element of the list should be an OPC leaf node. OPC servers can use the "." character in item names, so this hierarchy is not necessarily derivable from the point name.

**Returns**

A message indicating success or error. Please refer to *Return Syntax* for details.

**Description**

This command is deprecated in favor of *OPCAddItem2*, but is available to facilitate upgrading from earlier versions of the DataHub. It adds OPC items to an OPC server connection. It does not take effect until the *OPCApply* command is issued.
Example

(OPCAddItem "MyOPCServer" 2 0 "Tank3.Level" ("Tank3" "Level"))
**OPCAttach**

**OPCAttach** — is deprecated in favor of **OPCAttach2**.

**Synopsis**

```plaintext
(OPCAttach label machine_name interface_name server_pattern domain point_pattern [deadband_msec])
```

**Arguments**

- `label`  
  A name for this connection, as a string.

- `machine_name`  
  The name or IP address of the computer running the DataHub.

- `interface_name`  
  Is either: "OPC Data Access 2.0" or "OPC Data Access 3.0" Currently only "OPC Data Access 2.0" is supported.

- `server_pattern`  
  The name or IP address of the server computer. Wildcard characters are allowed.

- `domain`  
  The data domain name.

- `point_pattern`  
  A point name filter. Wildcard characters are allowed. Only leaf items that match this pattern will be visible in the DataHub. Normally you should specify * as the pattern.

- `deadband_msec`  
  The maximum rate at which the server should send data to the client.

**Returns**

A message indicating success or error. Please refer to Return Syntax for details.

**Description**

This command is deprecated in favor of **OPCAttach2**, but is available to facilitate upgrading from earlier versions of the OPC DataHub. It lets you set up an OPC connection. It corresponds roughly to the **Define OPC Server** dialog box that you get when you click the **Add** button in the **OPC** option of the Properties window.
OPCInit

OPCInit — is deprecated.

Synopsis

(OPCInit)
point

point — is used internally.

Synopsis

(point name type value [conf security locked seconds nanoseconds [quality]])

Arguments

name
The name of the point.

type
The type of the point.

value
The value of the point.

conf
The confidence level of the point.

security
The security level of the point.

locked
The locked status of the point.

seconds
The current time in seconds.

nanoseconds
The number of nanoseconds past seconds.

quality
The quality of the point.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is the string that the DataHub sends to all clients when a point value changes (i.e. a point exception occurs). The DataHub also listens for a point command because this is the mechanism that it uses to get updates from another DataHub that it is tunnelling/mirroring. This is not a command that a user would normally emit. Point changes should be sent to the DataHub using the write or cwrite commands.
qnx_name_attach

_qnx_name_attach_ — does nothing.

**Synopsis**

```
(qnx_name_attach node name)
```
qnx_receiver

qnx_receiver — does nothing.

Synopsis

(qnx_receiver name)
readid

readid — should not be used.

Synopsis

(readid pointnumber)

Arguments

pointnumber

The n\textsuperscript{th} point in the sender's default domain.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is like read but not as useful or robust. It reads the n\textsuperscript{th} point in the point table of the sender's default domain. This is neither useful nor not robust, since deleting a point will cause unpredictable behavior. Avoid using this command.
register_datahub

register_datahub — replaced by report_domain.

Synopsis

{(register_datahub domain)}
report_all

report_all — replaced by report_domain.

Synopsis

(report_all future domain_needed)
report_datahubs

report_datahubs — does nothing.

Synopsis

(report_datahubs yes|no)
request

request — replaced by report_domain.

Synopsis

(request pointname)
run

run — does nothing.

Synopsis

(run command [argument...])
**script_register**

*script_register* — is for internal use only.

**Synopsis**
Synopsis

— is for internal use only.
slave

slave — is for internal use only.

Synopsis
sync

sync — is for internal use only.

Synopsis
taskdied

taskdied — is for internal use.

Synopsis

(taskdied name domn qu nodename node pid chid)
taskstarted

— is for internal use.

Synopsis

(taskstarted name domn qu nodename node pid chid)
using_license

using_license — is for internal use only.

Synopsis
**warn_of_license_expiry**

*warn_of_license_expiry* — is deprecated.

**Synopsis**

```
(warn_of_license_expiry 0|1)
```
Appendix A. Command Line Options

For more control of the Cogent DataHub on startup, you can use its command line options. These options let you specify data domains, ports, and several other configuration items each time the DataHub starts. Except where noted, these options apply to the DataHub in Windows, Linux, or QNX, as well as Cascade Connect. You need to enter these options in the Properties section of the DataHub shortcut icon.

1. Right click on the DataHub shortcut icon and select Properties. The Cogent DataHub Properties window will appear, with the Shortcut tab selected:

2. Enter the options in the Target field, after the quotation marks, as illustrated above. The available options are as follows:

   -a
   Transmit all point messages to all registered clients, even if the value does not change.

   -b size
   The maximum message buffer size.

   -d domain
   The domain name for this DataHub. This option can be used multiple times to get multiple domains on a single DataHub.

   -D
   Do not detach from the controlling tty. Normally the DataHub will detach itself and become immune to interrupts and termination on the controlling tty. If this option is used, then an & is necessary to run datahub in the background.

   -f file
   Load this configuration file.

   -h
   Print a help message showing a summary of all these arguments.

   -H home_path
   The full path to the directory that will contain the configuration and license files.
This takes precedence over \texttt{-U}. If the directory cannot be found or created, the files will be stored in the installation directory.

\texttt{-I}

Hide the system tray icon when the DataHub starts. Only works in Windows.

\texttt{-l \textit{file}}

Log messages to this file.

\texttt{-m \textit{port}}

Acting as a TCP slave, attach to a TCP master on this port or service. The port is the matching port number of the master, usually 4502;

\texttt{-M \textit{address}}

Acting as a TCP slave, attach to a TCP master on this host. The address is a machine name, such as developers.cogentrts.com or a machine address, such as 192.168.3.15.

\texttt{-n \textit{domain}}

Acting as a TCP slave, tunnel/mirror this domain from the TCP master. The named domain on the master will be tunneled/mirrored to a domain of the same name in the slave.

\texttt{-p \textit{port}}

Act as a TCP master and listen on this port/service.

\texttt{-P}

Show the properties window when the DataHub starts. This is on by default in the desktop icon and Start menu entry.

\texttt{-q \textit{queue}}

Specify an alternate queue name for this DataHub. Normally \texttt{datahub} chooses its own queue name to be unique on the network.

\texttt{-s}

Synchronized: The DataHub will ignore changes to a point if the point's current timestamp is more recent.

\texttt{-t}

Automatically generate a timestamp on unstamped points.

\texttt{-U}

The DataHub should NOT create a directory within the user’s personal Application Data directory to store the configuration and license files, but rather in the application installation directory. This has a lower precedence than \texttt{-H}.

\texttt{-v}

Generate copious debugging information to the standard output. (Implies use of –
D)  
-\(V\)

Print the version number.

-\(X\)

Exit immediately (usually used with \(-V\)).

3. Click **OK** and restart the Cogent DataHub. The options you have chosen should take effect. Keep in mind that if your configuration file has different values for these options, it will override what you have entered here.
Appendix B. Excel Macro Library

This set of Excel macros each work on a 100 row x 40 column table of data in Sheet1 of the worksheet, starting at cell position A1. We have tried to make these macros generic so you can easily modify them to suit your needs.

Configure Excel to receive data from the Cogent DataHub (using DDEAdvise)

These macros normally need to be run only once, when first setting up a spreadsheet to receive data.

- **Attach array data in the Cogent DataHub, one array per row, to a table of values in Excel.** It is often more convenient to transmit large sets of Excel data as an array because this significantly reduces the bandwidth requirements and increases the speed of transmission. This macro sets up DDEAdvise loops from the DataHub to Excel, so that each row of the table is linked to an array point in the DataHub.

  ```vba
  Sub register_arrays()
      Dim pname As String
      For i = 1 To 100
          pname = Format(i, "0000")
          pname = "=datahub|default!array" & pname
          Worksheets("Sheet1").Range(Cells(i, 1), Cells(i, 40)).FormulaArray = pname
      Next i
  End Sub
  ```

  Each data point represents a row of data. This macro assumes the names are "array0001", "array0002", etc.

- **Attach individual point data in the Cogent DataHub, one point per cell, to a table of values.** This macro sets up DDEAdvise loops from the DataHub to Excel, so that each cell in the table is linked to a point in the Cogent DataHub.
This macro assumes that the data points are named "point0001", "point0002", etc.

Sub register_points()
    Dim pname As String
    For i = 1 To 100
        For j = 1 To 40
            pname = Format((i - 1) * 40 + j, "0000")
            pname = "=datahub\default!point" & pname
            Worksheets("Sheet1").Cells(i, j).Formula = pname
        Next j
    Next i
End Sub

Write data from Excel - User initiated (using DDEPoke)

These macros are useful for writing data out from Excel on demand. In other words, the user decides when to write the data, and does so by running one of these macros (usually from an assigned button click).

• Transmit array data, one array per row, to points in the Cogent DataHub. Triggering this macro writes all the data from the table in Excel to the Cogent DataHub. The macro writes each row of the table as an array point in the DataHub. All rows of the table get transmitted, one after another.

This macro assumes that the data points are named "array0001", "array0002", etc.

Sub transmit_arrays()
    Dim chan As Integer
    Dim pname As String

    chan = DDEInitiate("datahub", "default")
    For i = 1 To 100

Transmit individual point data, one point per cell, to points in the Cogent DataHub. Triggering this macro writes all the data from the Excel table to the DataHub. The macro writes each cell of the table in turn to a single point in the DataHub.

```
Sub transmit_points()
    Dim chan As Integer
    Dim pname As String
    chan = DDEInitiate("datahub", "default")
    For i = 1 To 100
        For j = 1 To 40
            pname = Format((i - 1) * 40 + j, "0000")
            pname = "point" & pname
            DDEPoke chan, pname, Worksheets("Sheet1").Cells(i, j)
        Next j
    Next i
    DDETerminate (chan)
End Sub
```

This macro assumes that the data points are named "point0001", "point0002", etc.

Write data from Excel - Automatically on value change (using DDE-Poke)

These macros are useful for automatically transmitting data from Excel into the Cogent DataHub.

- **Emit new cell values to the Cogent DataHub.** Whenever a user enters a new value, this macro checks to see if that cell is named. If so, the macro emits the new value to a Cogent DataHub point of the same name. The subroutine name "Worksheet_Change" is special - it is called by Excel whenever a change occurs on the Worksheet due to user
input or recalculation (though not a change due to a DDE message; for that see Other Useful Macros).

---

Sub Worksheet_Change(ByVal Target As Range)
    Dim rname As String
    Dim channel As Variant
    On Error Resume Next
    rname = Target.name.name
    If Not rname = "" Then
        channel = DDEInitiate("datahub", "default")
        DDEPoke channel, rname, Target
        DDETerminate (channel)
    End If
End Sub
---

• Transmit changes to a range. This pair of macros determines that a cell within a particular named range has changed through user input, and transmits the contents of the range to the Cogent DataHub. This is useful because you do not have to configure each cell you want to write out to the DataHub. If the cell that is changed lies within a defined range, then all values in that range are automatically written out to the DataHub.

The Worksheet_Change Routine determines the enclosing range for the change, and if the range matches one of a predefined set, it will send that range to the Cogent DataHub. The NameOfParentRange function determines the name of the cell range that intersects a given range. If more than one named range in the worksheet intersects the given range, it returns only the first one.

Add to Workbook Macro Code:

---

Function NameOfParentRange(Rng As Range) As String
    Dim Nm As Name
    For Each Nm In ThisWorkbook.Names
        If Rng.Parent.Name = Nm.RefersToRange.Parent.Name Then
            If Not Application.Intersect(Rng, Nm.RefersToRange) Is Nothing Then
                NameOfParentRange = Nm.Name
                Exit Function
            End If
        End If
    Next Nm
    NameOfParentRange = ""
End Function
---

Add to Sheet1 macro code:
space
Excel Macro Library

space
657

--------------------------------------------------Sub Worksheet_Change(ByVal r As Range)
Dim pname As String
Dim chan As Integer
pname = ThisWorkbook.NameOfParentRange(r)
If Not pname = "" Then
On Error Resume Next
chan = DDEInitiate("datahub", "default")
DDEPoke chan, pname, Worksheets("Sheet1").Range(pname)
DDETerminate (chan)
End If
End Sub
---------------------------------------------------

Other Useful Macros
• Cause a macro to run when a Cogent DataHub point changes value. Here is one
macro that runs another macro every time a certain cell's value is updated by a DDE
message. The macro that gets run is link_updated. It simply increments the value in
cell A1. You can easily change this example to meet your needs. The set_link macro
tells the workbook to run the link_updated macro whenever the DataHub sends a
DDE message about point0001. You can also change the name of the DataHub point
as needed.

Add to Sheet1 macro code:
--------------------------------------------------Sub link_updated()
Cells(1, 1) = Cells(1, 1) + 1
End Sub
--------------------------------------------------Run once to establish the link:
--------------------------------------------------Sub set_link()
ThisWorkbook.SetLinkOnData "datahub|default!'point0001'", _
"Sheet1.link_updated"
End Sub
---------------------------------------------------

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com


Appendix C. Running as a Windows Service (Specified User)

The following information relates to running the DataHub service as a specified user. We recommend not doing this as the properties are only available on the service console of the SYSTEM user account. But in some cases this may be unavoidable.

![Configure Cogent DataHub Service](image)

If you try to install the DataHub service as a specified user as explained in the section called “Installing the DataHub as a Service” then you may find that the service does not run, and you get an error message related to a logon failure, such as this:

![Error Message](image)

This error occurs when a user who is going to run the DataHub service does not have the permission, or right, to log on as a service in Windows.

One way to set this permission/right is as follows:

1. Open the Windows **Services** window from the **Control Panel** by selecting **Administrative Tools** and then **Services**.
2. Right click on **Cogent DataHub** and select **Properties**.
3. Click on the Log On tab and then retype the password for the user account you are trying to use. Then click Apply.

4. You should see a Windows message acknowledging that the right to log on as a service has been granted.

5. Now if you try to start the service from the Cogent DataHub Services Manager again it should work.
Appendix D. Windows Services File for Tunnel/Mirror

If you enter a name for the Master service/port in the Tunnel/Mirror tab of the Properties window, that name must be listed in the Windows services file.

Finding the services file

Since each manufacturer's services file is different, you must find the services file that your TCP/IP protocol stack is currently using. A Microsoft TCP/IP implementation typically puts the services file in the C:\Windows (or equivalent) directory. Most third party software either installs the services file in the same directory that their software was installed or into a directory named C:\ETC. Refer to your TCP/IP documentation for the location of this file.

- In Windows NT, the installation program attempts to edit a services file in the \win\nt\system32\drivers\etc directory.
- In Windows XP, the installation program currently does not attempt to edit the services file. The default directory for that file is C:\WINDOWS\system32\drivers\etc .

Editing the services file

Once you have found the services file, you must add the line:

```
service_name ####/TCP
```

using a text editor. For example, to assign the name datahub to port 4502, you would add the line:

```
datahub 4502/TCP
```

Remember that if you edit the services file with Notepad, it will attach a .txt suffix when it saves the file so that you will not in fact have edited the system services file, but instead created a new file, named services.txt. You should rename that file services, without the .txt extension.
Appendix E. DataHub Registry Entries

The DataHub places registry entries in HKEY_CURRENT_USER\Software\Cogent\Cogent DataHub. Many of those entries are used by the DataHub to store window position and size. Some of these are user-modifiable to adjust the DataHub’s behaviour. The following table contains the user-modifiable entries. If an entry does not exist, then the user can create it to achieve the effect. In most cases the DataHub must be restarted for the registry entry to apply.

<table>
<thead>
<tr>
<th>Registry Value Name</th>
<th>Type</th>
<th>Effect</th>
</tr>
</thead>
</table>
| NetworkComputerPoll             | DWORD| Controls the polling for network computer names. By default the DataHub will poll the network periodically for the Windows names of all computers. This may cause unwanted network traffic. Setting this value will have the following effect:  
• 0 = Do not poll for network computer names  
• 1 = Poll for network computer names exactly once when the DataHub starts  
• 1000 or higher = Poll periodically with a base polling rate of this many milliseconds  
Numbers < 0 are treated as 0. Numbers between 2 and 1000 are treated as 1000. If the key is absent then the DataHub defaults to 10,000ms. |
| EmitConsoleEvents               | DWORD| Cause the Event Log to be emitted to the debugger console. This is only useful if the DataHub is running within a debugger (Visual Studio or WinDbg).  
• 0 = Do not emit the Event Log to the debug console.  
• 1 = Emit the Event Log to the debug console. |
| EmitConsoleDebug                | DWORD| Cause debug-level messages to be emitted to the debugger console. This will only take effect if EmitConsoleEvents is also enabled.  
• 0 = Do not emit debug-level events to the debugger console.  
• 1 = Emit debug-level events to the debugger console. |
<table>
<thead>
<tr>
<th>Registry Value Name</th>
<th>Type</th>
<th>Effect</th>
</tr>
</thead>
</table>
| StatusDomainVisible            | DWORD  | The DataHub maintains an internal data domain called “Status” that contains information about its state. This domain is normally not visible in the data viewer, nor visible in queries of available data domains. This value will make the Status domain visible.  
  - 0 = Status domain is not visible.  
  - 1 = Status domain is visible. |
| MainThreadPriority             | DWORD  | Sets the operating system thread priority for the main DataHub thread. |
| TimerResolution                | DWORD  | Sets the maximum timer resolution in milliseconds. The DataHub will adjust the system “multimedia timer” to this resolution. If this value does not exist then the DataHub will query the operating system for the highest available resolution and use that. This number must be at least 1. |
| AdjustTimer                    | DWORD  | Enables multimedia timer adjustment. If this value exists and is not zero then the DataHub will adjust the system-wide multimedia timer resolution to the value described in TimerResolution.  
  - 0 = Do not adjust the system multimedia timer.  
  - 1 = Adjust the system multimedia timer.  
  The default is 1. |
| DDEAllowBusyAdvise             | DWORD  | Allow the DataHub to process a DDE Advise message while another DDE message is being processed. This is generally dangerous, as the DDE protocol can become confused over which messages belong to a specific transaction. Enabling this could result in failed DDE transactions.  
  - 0 = Do not allow DDE Advise while another DDE message is in process.  
  - 1 = Allow DDE Advise while another DDE message is in process.  
  The default is 0. |
<table>
<thead>
<tr>
<th>Registry Value Name</th>
<th>Type</th>
<th>Effect</th>
</tr>
</thead>
</table>
| SslMethod                 | String | Force the DataHub to use a specific SSL method. If this registry entry does not exist then the DataHub will use the best available method. Can be one of:  
  • SslV23  
  • SslV3  
  • Tls  
  • Tls1.1  
  • Tls1.2 |
| EventLogFontSize          | DWORD  | The font size of the Event Log window.                                  |
| EventLogDirectWrite       | DWORD  | Event Log display technology selection. This can be one of:             |
|                           |        |  • 0 = default  
  • 1 = DirectWrite  
  • 2 = DirectWrite Retain  
  • 3 = DirectWrite to 2D device context. |
| EventLogFont              | String | The name of the font to use in the Event Log window.                    |
| TCPLicenseTimeoutSecs     | DWORD  | The DataHub waits for 30 seconds after a client connects before verifying the client's license. If the client does not transmit licensing information within the 30 second window, the DataHub will assume that the connection is from a custom application using a direct TCP connection, and will attempt to assign a TCP Link license to that client. If no TCP Link license is available, the DataHub will terminate the connection with a "no license" error.  
  
  You can modify this licensing window up to 60 seconds by creating this DWORD registry entry.  
  
  If you are attempting to tunnel data on a very slow network, it is possible that the slave (client) DataHub sends its licensing information within the 30 second window, but the master DataHub does not receive it in time. In this case the DataHub will report a no license error even though both the slave and master are correct- |
<table>
<thead>
<tr>
<th>Registry Value Name</th>
<th>Type</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCPMaxQueuedBytes</td>
<td>DWORD</td>
<td>The number of bytes of output that a TCP socket can buffer before the socket is considered “hung” as closed. The default is 50,000,000.</td>
</tr>
<tr>
<td>TCPHungQueuedBytes</td>
<td>DWORD</td>
<td>The number of bytes of output that a TCP socket can buffer before the socket starts refusing new data point updates. This is a throttling mechanism that results in old data point values being dropped when newer values for the same data point become available. When the socket buffer is cleared then the DataHub automatically stops throttling. The default is 200,000.</td>
</tr>
</tbody>
</table>
| DisableTCPKeepalive  | DWORD | • 0 = Enable TCP keepalive messages.  
• 1 = Disable TCP keepalive messages.  
The default is 0. |
| BrowseDA3            | DWORD | Force the OPC DA Classic item browser to use a specific technology (OPC DA2 or DA3). If this value does not exist then browsing will use the selected technology from the OPC client configuration dialog.  
• 0 = Use OPC DA2 when browsing items.  
• 1 = Use OPC DA3 when browsing items. |
<p>| OPCClientGroupName   | String| Normally the DataHub will set its OPC DA client group name to the program name. If this registry entry exists then this value will be used as the OPC DA client group name instead. |
| OPCClientSequentialGroups | DWORD | This entry tells the DataHub to append a sequence number to the OPC client group name. This can be useful if you are tracking an issue with repeated DCOM disconnections. |</p>
<table>
<thead>
<tr>
<th>Registry Value Name</th>
<th>Type</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• 0 = Do not add a sequence number to the group name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 = Add a sequence number to the group name.</td>
</tr>
<tr>
<td>OPCClientGroupName</td>
<td>DWORD</td>
<td>This entry sets the current sequence number for the OPC client group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It persists through starts and stops of the DataHub.</td>
</tr>
<tr>
<td>OPCCheckMemory</td>
<td>DWORD</td>
<td>This entry enables an additional memory check for some OPC DA operations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This option increases CPU usage substantially, so only use this if you suspect an OPC-related memory problem.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0 = Do not enable additional memory checks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 = Enable additional memory checks.</td>
</tr>
<tr>
<td>OpcAeSkipAttributes</td>
<td>DWORD</td>
<td>This entry will instruct all OPC A&amp;E client (outbound) connections to ignore condition attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This can save memory when there are a large number of conditions with attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0 = Retain condition attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 = Ignore condition attributes.</td>
</tr>
</tbody>
</table>
Appendix F. OPC Overview

OPC is a software interface standard that allows HMIs and other programs to communicate with industrial hardware devices.

The acronym "OPC" originally came from "OLE for Process Control". Since OLE (Object Linking and Embedding) is based on the Windows COM (Component Object Model) standard, under the hood OPC was essentially COM. Due to limitations of the COM standard related to networking, security, and reliance on the Windows operating system, the OPC Foundation introduced OPC UA (for "Universal Architecture"), and renamed the original OPC "OPC Classic." They also disassociated the "OPC" acronym from any particular meaning.

OPC Classic comprises several standards, the first and most important of which is OPC Data Access (DA). There are also standards for Alarms & Events (A&E), Historical Data Access (HDA), and others. OPC UA implements these types of functionality, while running on both Windows and non-Windows operating systems, and providing better networking support and a more sophisticated security model than OPC Classic. Nevertheless, OPC Classic has been highly successful for many years, is widely available, and has a very large installed user base. For these reasons, both OPC Classic and OPC UA are expected to be popular industrial data communications protocols for some time to come.

OPC is implemented in server/client pairs. The **OPC server** is a software program that converts the hardware communication protocol used by a PLC\(^1\) into the OPC protocol. The **OPC client** software is any program that needs to connect to the hardware, such as an HMI\(^2\). The OPC client uses the OPC server to get data from or send commands to the hardware.

The value of OPC is that it is an open standard, which means lower costs for manufacturers and more options for users. Hardware manufacturers need only provide a single OPC server for their devices to communicate with any OPC client. Software vendors simply include OPC client capabilities in their products and they become instantly compatible with thousands of hardware devices. Users can choose any OPC client software they need, resting assured that it will communicate seamlessly with their OPC-enabled hardware, and vice-versa.

---

\(^1\)Programmable Logic Controller: a small industrial computer that controls one or more hardware devices.

\(^2\)Human-Machine Interface: a graphical interface that allows a person to interact with a control system. It may contain trends, alarm summaries, pictures, or animation.
The typical OPC connection scenario is a single server-client connection on a single computer as illustrated above, but there are more possibilities. For example, you might need to:

- Connect an OPC client to several OPC servers. This is called aggregation.
- Connect an OPC Classic DA client to an OPC Classic DA server over a network. This can be done with tunnelling.
- Connect an OPC server to another OPC server to share data. This is known as bridging.

The Cogent DataHub is uniquely designed to do all of these tasks, and more. It combines OPC server and OPC client functionality (OPC Classic DA and A&E, as well as UA) to support multiple connections. Thus it can connect to several OPC servers simultaneously, for aggregation and bridging. Two DataHubs can mirror data across a TCP network to provide tunnelling, typically used for OPC Classic DA. And because it supports both OPC Classic and OPC UA, the DataHub can be used as a protocol converter between these two.

In addition to enhancing OPC server and client connections, the DataHub can connect any OPC server or client to other applications as well, such as Excel, a web browser, or any ODBC database.
Appendix G. DDE Overview

DDE (Dynamic Data Exchange) is a well-established mechanism for exchanging data among processes in MS-Windows. The mechanism was intentionally designed to be easy to use and to represent data as simply as possible. DDE is implemented in many popular programs that run in Windows, such as Microsoft Excel and Microsoft Word. This widespread availability makes DDE a good choice for general data sharing.

The competition with DDE is COM, with its variants for OLE: OPC and ActiveX. By comparison, DDE is simpler, and therefore faster, than the equivalent COM interface if implemented as a separate process. DDE is much easier to implement in code, and offers a particular data model as (name, value) pairs. In the case of real-time data, this model is well suited, and therefore offers the best cost/benefit ratio when programming for real-time data.

However, DDE was not designed to be used over a network. The Cogent solution for this shortcoming is to tunnel/mirror two copies of the Cogent DataHub over a network or the Internet using TCP. Thus, two programs that use only DDE can exchange data across a robust, TCP-enabled link.

Data Definitions

DDE defines data in terms of (service, topic, item), explained as follows:

- **service**
  A name used by a DDE server to identify its service to DDE clients. The default DDE service name for the Cogent DataHub is `datahub`. Unlike most Windows programs, the Cogent DataHub lets you change this name, or add more names if you’d like.

- **topic**
  A way to categorize items. This corresponds to a Cogent DataHub `data domain`.

- **item**
  A variable that holds a value. This corresponds to a Cogent DataHub `point`.

Here are some service and topic names for several Windows programs:

<table>
<thead>
<tr>
<th>Application</th>
<th>DDE Service Name</th>
<th>DDE Topic Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cogent DataHub</td>
<td>Multiple names can be assigned. The default name is <code>datahub</code>.</td>
<td>Any Cogent DataHub domain name, the default domain is <code>default</code>.</td>
</tr>
<tr>
<td>Microsoft Excel</td>
<td>EXCEL</td>
<td>The name of the spreadsheet, chart, macro, etc. For example: <code>mysheet.xls</code></td>
</tr>
<tr>
<td>Microsoft Access</td>
<td>MSACCESS</td>
<td>The name of the table, SQL query, or macro to run.</td>
</tr>
</tbody>
</table>
### DDE Overview

<table>
<thead>
<tr>
<th>Application</th>
<th>DDE Service Name</th>
<th>DDE Topic Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Word</td>
<td>WINWORD</td>
<td>The name of the document, including the .doc extension.</td>
</tr>
<tr>
<td>FoxPro</td>
<td>FOXPRO</td>
<td></td>
</tr>
<tr>
<td>InTouch Viewer</td>
<td>VIEW</td>
<td>TAGNAME</td>
</tr>
<tr>
<td>FIX DMACS</td>
<td>DMDDDE</td>
<td>DATA</td>
</tr>
<tr>
<td>National Instruments' Lookout</td>
<td>LOOKOUT</td>
<td>The name of the application, without the .lkp extension.</td>
</tr>
<tr>
<td>Asymetrix Toolbook</td>
<td>TOOLBOOK</td>
<td>The name of the toolbar application, with the .tbk extension.</td>
</tr>
</tbody>
</table>

Here are a few service and topic names for financial data feeds:

<table>
<thead>
<tr>
<th>Data Feed</th>
<th>DDE Service Name</th>
<th>DDE Topic Name</th>
<th>Symbol Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADP Shark</td>
<td>ML</td>
<td>LP</td>
<td>IBM.N.Q</td>
</tr>
<tr>
<td>Bloomberg</td>
<td>BLP</td>
<td>M</td>
<td>IBM EQUITY,[LAST TRADE]</td>
</tr>
<tr>
<td>Bridge</td>
<td>BDDE</td>
<td>TKR</td>
<td>@USH8/LS</td>
</tr>
<tr>
<td>FXCM</td>
<td>FXPS</td>
<td>BID</td>
<td>EUR/USD</td>
</tr>
<tr>
<td>Future Source (CalcSource)</td>
<td>CALCSRC</td>
<td>P</td>
<td>USH8.LAST</td>
</tr>
<tr>
<td>Future Source (ProfNet)</td>
<td>PROFDDE</td>
<td>LIVE</td>
<td>APIU02,LAST</td>
</tr>
<tr>
<td>Knight Rider Profit Center</td>
<td>QMASTER</td>
<td>QUOTE</td>
<td>USH8.LAST</td>
</tr>
<tr>
<td>METATRADER</td>
<td>MT</td>
<td>BID</td>
<td>USDCHF</td>
</tr>
<tr>
<td>METATRADER v. 4</td>
<td>MT4</td>
<td>BID</td>
<td>USDCHF</td>
</tr>
<tr>
<td>MGFOREX</td>
<td>MGFOREX</td>
<td>RATES</td>
<td>USD</td>
</tr>
<tr>
<td>Moneycast</td>
<td>WBSERVER</td>
<td>SES</td>
<td>19/L (19 is a stock code)</td>
</tr>
<tr>
<td>Reuters</td>
<td>REUTER</td>
<td>IDN</td>
<td>USH8 /IBM ,LAST</td>
</tr>
<tr>
<td>Telerate WorkStation</td>
<td>TWINDDE</td>
<td>QUOTES</td>
<td>USH8.3 LAST</td>
</tr>
<tr>
<td>Universal Market Data Server</td>
<td>USDDE</td>
<td>QUOTE</td>
<td>US8H;LAST</td>
</tr>
</tbody>
</table>

### Client and Server

In DDE, the role of client and server in data exchange is fairly clear. A client initiates the activity, and the server responds. To facilitate two-way data transfer, each Cogent DataHub can each act as both client and server. This is what allows Excel spreadsheets to share data bidirectionally across a network.
Sending and Receiving Data

There are three ways to send and receive ordinary data over DDE. Here is a brief explanation, using the Cogent DataHub and Microsoft Excel as examples:

**Poke** The client sends data for an item directly to the server. In Excel, this is done with a macro. The server does not necessarily reply. The actual data flow is from client to server—from Excel to DataHub.

**Request** The client asks the server to send an item's data. In Excel, this is done with a macro. The client receives either the requested value, or NULL if the server can't send the value. The actual data flow is from server to client—from DataHub to Excel.

**Advise** The client asks to be notified of any change in the data for an item. If the server agrees to the request, it sends the new value for the item each time its value changes. The Cogent DataHub can conduct two-way communication with Excel using only this advise capability, as follows:

- To receive data into Excel from the DataHub, you set the DataHub to act as a DDE server, which requires you to enter a service name. This identifies the DataHub to Excel. Then you can drag and drop a point name from the DataHub into a cell of an Excel spreadsheet.
- To send data from Excel to the DataHub, you set the DataHub to act as a DDE client, which requires you to enter service, topic, and item names. These identify the item in the spreadsheet to the DataHub. Then, in Excel, you name a cell with the DataHub point name.
Appendix H. ODBC Database Concepts

Database Terminology and Concepts

In general, a *database* is a collection of information. Computerized databases store data in *tables*, which are accessed through a *database management system* or DBMS, such as SQL Server, MS Access, MySQL, Oracle, etc. All of these are capable of storing data in related, linked tables, which taken together are called a *relational database*. Most modern computerized databases are relational databases.

A database *table* is a logical grouping of related data, organized by the common attributes or characteristics of individual data items. Each *column* or *field* in the table contains a particular attribute, and is of one particular *data type*. Typical data types include boolean, string, numeric, date/time, etc. Each *row* or *record* in the table contains a complete set of every *data value* related to a single item.

For example, a table containing data from the Cogent DataHub might have columns (fields) for a point name, value, timestamp, and quality. Each row (record) would show the various data values logged for that point at different times:

![Table Example]

A different kind of table might have one column for a timestamp, and then additional columns containing the values of different DataHub points logged at each time, like this:

![Another Table Example]

A database table may require each row (or record) to be uniquely identified. This is commonly done through a *key column*, whose main and (sometimes only) purpose is to provide a unique identifying value to the row. Most DBMSs allow this value to be assigned manually by the database user, or automatically through an *auto-incrementing counter* or other mechanism. It is possible for a table to have multiple key columns, but some functions in the Cogent DataHub will only write to tables with a single key column. For more in-
information about configuring database tables with the Cogent DataHub, please refer to the section called “Configuring a Database Table”

**Connecting to a Database: ODBC**

Connecting to a database is done through the DBMS, which normally offers two possibilities: *native drivers* and *ODBC* (Open Database Connectivity). Native drivers are inconvenient to use because each requires its own programming interface. ODBC, on the other hand, specifies a standardized, common interface that is available from almost every database vendor, including SQL Server, MS Access, MySQL, Oracle, and many, many more. The Cogent DataHub uses ODBC to connect to databases.

ODBC supports communication with a DBMS locally or across a network, using an *ODBC driver*. Every ODBC-compliant DBMS provides an ODBC driver, which needs to be installed on the user’s machine. For example, there is an ODBC driver for MS Access, for SQL Server, for MySQL, and so on.

You can use the Windows **ODBC Data Source Administrator** to configure a connection between the ODBC driver and the specific database you want to work with. That configuration is called the *Data Source Name*, or *DSN*. For example, the Cogent DataHub references the DSN and uses the configured connection for the ODBC driver to connect to the database.

Configuring the DSN is straightforward, varying slightly depending on the ODBC driver you are working with. Usually you need to select an ODBC driver, create a name for the DSN, and select a database. Other information, such as a login name or password may be required or optional. For more information, please refer to the section called “Setting up the DSN (Data Source Name)”

**User Name and Password**

According to Microsoft: "If used in an OLE DB or ODBC connection string, a login or password must not contain the following characters:

```
[] {} () , ; ? * ! @
```

These characters are used to either initialize a connection or separate connection values."
In addition to this, the single and double quote characters (' and ") may not work, depending on the database.

**Accessing Data: SQL**

Once connected to a database, any queries (requests) to retrieve, modify, add, or delete data must be made through a language. The most popular database query language is SQL (Structured Query Language), pronounced "sequel" or "ess-kyu-el". Created in the 1960s, this language has become a widely-used standard supported by most DBMSs, although there are some minor variations in certain commands offered.

The Cogent DataHub uses SQL to write to and read from databases. When you configure the Data Logging interface to write DataHub point values, under the hood the commands used are written in SQL. The DataHub scripts also use SQL commands to write and read data. For example, the following line from the `ODBCTutorial13.g` script uses an SQL `SELECT` command to pull all of the data from a database table.

```sql
result = .conn.QueryToClass (.tableclass, 
  string ("select * from ", .tablename));
```

The `SELECT` command is often used with the `FROM` and `WHERE` operators, in queries such as this:

```sql
SELECT data_element_1 
FROM table_1 
WHERE data_element_1 > 32 and data_element_1 < 212;
```

The syntax of SQL is fairly simple, and there are many books and online tutorials that can help you learn. The information presented here about SQL, ODBC, and databases in general should be enough to get started logging data with the Cogent DataHub.
# Appendix I. Error Messages

This section presents error numbers that the developer may encounter when using the Cogent DataHub.

## Windows Error Numbers

<table>
<thead>
<tr>
<th>Number</th>
<th>Error String</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EOK</td>
<td>No error.</td>
</tr>
<tr>
<td>1</td>
<td>EPERM</td>
<td>No permissions, or the user is not the process owner.</td>
</tr>
<tr>
<td>2</td>
<td>ENOENT</td>
<td>No such file or directory.</td>
</tr>
<tr>
<td>3</td>
<td>ESRCH</td>
<td>No such process.</td>
</tr>
<tr>
<td>4</td>
<td>EINTR</td>
<td>Interrupted system call.</td>
</tr>
<tr>
<td>5</td>
<td>EIO</td>
<td>I/O error.</td>
</tr>
<tr>
<td>6</td>
<td>ENXIO</td>
<td>No such device or address.</td>
</tr>
<tr>
<td>7</td>
<td>E2BIG</td>
<td>Argument list is too big.</td>
</tr>
<tr>
<td>8</td>
<td>ENOEXEC</td>
<td>Executable format is not recognized.</td>
</tr>
<tr>
<td>9</td>
<td>EBADF</td>
<td>Bad file number or invalid file descriptor.</td>
</tr>
<tr>
<td>10</td>
<td>ECHILD</td>
<td>No child processes exist.</td>
</tr>
<tr>
<td>11</td>
<td>EAGAIN</td>
<td>Resource temporarily unavailable or operation would block.</td>
</tr>
<tr>
<td>12</td>
<td>ENOMEM</td>
<td>Out of memory.</td>
</tr>
<tr>
<td>13</td>
<td>EACCES</td>
<td>Permission denied.</td>
</tr>
<tr>
<td>14</td>
<td>EFAULT</td>
<td>Bad memory address.</td>
</tr>
<tr>
<td>15</td>
<td>ENOTBLK</td>
<td>Block operation attempted on non-block device.</td>
</tr>
<tr>
<td>16</td>
<td>EBUSY</td>
<td>Device or resource busy, or operation already in progress.</td>
</tr>
<tr>
<td>17</td>
<td>EEXIST</td>
<td>File exists.</td>
</tr>
<tr>
<td>18</td>
<td>EXDEV</td>
<td>Cross-device link.</td>
</tr>
<tr>
<td>19</td>
<td>ENODEV</td>
<td>No such device.</td>
</tr>
<tr>
<td>20</td>
<td>ENOTDIR</td>
<td>Not a directory.</td>
</tr>
<tr>
<td>21</td>
<td>EISDIR</td>
<td>Is a directory.</td>
</tr>
<tr>
<td>22</td>
<td>EINVAL</td>
<td>Invalid argument.</td>
</tr>
<tr>
<td>23</td>
<td>ENFILE</td>
<td>File table overflow.</td>
</tr>
<tr>
<td>Number</td>
<td>Error String</td>
<td>Error Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>24</td>
<td>EMFILE</td>
<td>Too many open files.</td>
</tr>
<tr>
<td>25</td>
<td>ENOTTY</td>
<td>Character operation on non-character device.</td>
</tr>
<tr>
<td>26</td>
<td>ETXTBSY</td>
<td>Text file is busy.</td>
</tr>
<tr>
<td>27</td>
<td>EFBIG</td>
<td>File is too large.</td>
</tr>
<tr>
<td>28</td>
<td>ENOSPC</td>
<td>No space left on device.</td>
</tr>
<tr>
<td>29</td>
<td>ESPIPE</td>
<td>Illegal seek attempted on a pipe.</td>
</tr>
<tr>
<td>30</td>
<td>EROFS</td>
<td>Attempted write to a read-only file system.</td>
</tr>
<tr>
<td>31</td>
<td>EMLINK</td>
<td>Too many links.</td>
</tr>
<tr>
<td>32</td>
<td>EPIPE</td>
<td>Broken pipe.</td>
</tr>
<tr>
<td>33</td>
<td>EDOM</td>
<td>Math argument out of data domain of function.</td>
</tr>
<tr>
<td>34</td>
<td>ERANGE</td>
<td>Result too large.</td>
</tr>
<tr>
<td>35</td>
<td>EUCLEAN</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>EDEADLOCK</td>
<td>Deadlock avoided.</td>
</tr>
</tbody>
</table>

**Windows TCP Error Numbers**

<table>
<thead>
<tr>
<th>Number</th>
<th>Error String</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10035</td>
<td>EWOULDBLOCK</td>
<td>Resource temporarily unavailable or operation would block.</td>
</tr>
<tr>
<td>10036</td>
<td>EINPROGRESS</td>
<td>Operation now in progress.</td>
</tr>
<tr>
<td>10037</td>
<td>EALREADY</td>
<td>Device or resource busy, or operation already in progress.</td>
</tr>
<tr>
<td>10038</td>
<td>ENOTSOCK</td>
<td>Socket operation on non-socket.</td>
</tr>
<tr>
<td>10039</td>
<td>EDESTADDRREQ</td>
<td>Destination address required.</td>
</tr>
<tr>
<td>10040</td>
<td>EMSGSIZE</td>
<td>Message too long.</td>
</tr>
<tr>
<td>10041</td>
<td>EPROTOTYPE</td>
<td>Protocol wrong type for socket.</td>
</tr>
<tr>
<td>10042</td>
<td>ENOPROTOOPT</td>
<td>Protocol not available.</td>
</tr>
<tr>
<td>10043</td>
<td>EPROTONOSUPPORT</td>
<td>Protocol not supported.</td>
</tr>
<tr>
<td>10044</td>
<td>ESOCKTNOSUPPORT</td>
<td>Socket type not supported.</td>
</tr>
<tr>
<td>10045</td>
<td>EOPNOTSUPP</td>
<td>Operation not supported.</td>
</tr>
<tr>
<td>10046</td>
<td>EPFNOSUPPORT</td>
<td>Protocol family not supported.</td>
</tr>
<tr>
<td>10047</td>
<td>EAFNOSUPPORT</td>
<td>Address family not supported by protocol family.</td>
</tr>
<tr>
<td>Number</td>
<td>Error String</td>
<td>Error Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>10048</td>
<td>EADDRINUSE</td>
<td>Address already in use.</td>
</tr>
<tr>
<td>10049</td>
<td>EADDRNOTAVAIL</td>
<td>Can't assign requested address.</td>
</tr>
<tr>
<td>10050</td>
<td>ENETDOWN</td>
<td>Network is down.</td>
</tr>
<tr>
<td>10051</td>
<td>ENETUNREACH</td>
<td>Network is unreachable.</td>
</tr>
<tr>
<td>10052</td>
<td>ENETRESET</td>
<td>Network dropped connection on reset.</td>
</tr>
<tr>
<td>10053</td>
<td>ECONNABORTED</td>
<td>Software caused connection abort.</td>
</tr>
<tr>
<td>10054</td>
<td>ECONNRESET</td>
<td>Connection reset by peer.</td>
</tr>
<tr>
<td>10054</td>
<td>ECONNRESET</td>
<td>Connection reset by peer.</td>
</tr>
<tr>
<td>10055</td>
<td>ENOBUFS</td>
<td>No buffer space available.</td>
</tr>
<tr>
<td>10056</td>
<td>EISCONN</td>
<td>Socket is already connected.</td>
</tr>
<tr>
<td>10057</td>
<td>ENOTCONN</td>
<td>Socket is not connected. Note: If this error appears with this message: TCP master service initialization failed: 10057 it could mean that a program is holding open port 4502. This is may be caused by the DataHub not shutting down properly for some reason and it is still running, even though the icon is not showing. To check, look in the Windows Task List and see if you can see a DataHub running. If it is, then you can kill it in the list and restart the DataHub. If it’s not in the list, then you need to use a firewall program to check to see which program is using port 4502.</td>
</tr>
<tr>
<td>10058</td>
<td>ESHUTDOWN</td>
<td>Can't send after socket shutdown.</td>
</tr>
<tr>
<td>10059</td>
<td>ETOMANYREFS</td>
<td>Too many references: can't splice.</td>
</tr>
<tr>
<td>10060</td>
<td>ETIMEDOUT</td>
<td>Connection timed out.</td>
</tr>
<tr>
<td>10061</td>
<td>ECONNREFUSED</td>
<td>Connection refused.</td>
</tr>
<tr>
<td>10062</td>
<td>ELOOP</td>
<td>Too many symbolic link or prefix loops.</td>
</tr>
<tr>
<td>10063</td>
<td>ENAMETOOLONG</td>
<td>Name too long.</td>
</tr>
<tr>
<td>10064</td>
<td>EHOSTDOWN</td>
<td>Host is down.</td>
</tr>
<tr>
<td>10065</td>
<td>EHOSTUNREACH</td>
<td>No route to host.</td>
</tr>
<tr>
<td>Number</td>
<td>Error String</td>
<td>Error Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>10066</td>
<td>ENOTEMPTY</td>
<td>Directory not empty.</td>
</tr>
<tr>
<td>10067</td>
<td>EPROCLIM</td>
<td>Process count limit reached.</td>
</tr>
<tr>
<td>10068</td>
<td>EUSERS</td>
<td>User count limit reached.</td>
</tr>
<tr>
<td>10069</td>
<td>EDQUOT</td>
<td>Quota limit reached.</td>
</tr>
<tr>
<td>10070</td>
<td>ESTALE</td>
<td>Potentially recoverable i/o error.</td>
</tr>
<tr>
<td>10071</td>
<td>EREMOTE</td>
<td>Too many levels of remote in path.</td>
</tr>
</tbody>
</table>

Windows DDE Error Numbers

<table>
<thead>
<tr>
<th>Number</th>
<th>Error String</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16384</td>
<td>DMLERRADVACKTIMEOUT</td>
<td>Timeout waiting for an advise acknowledge.</td>
</tr>
<tr>
<td>16385</td>
<td>DMLERRBUSY</td>
<td>Recipient is busy.</td>
</tr>
<tr>
<td>16386</td>
<td>DMLERRDATAACKTIMEOUT</td>
<td>Timeout waiting for an advise data acknowledge</td>
</tr>
<tr>
<td>16387</td>
<td>DMLERR_DLL_NOT_INITIALIZED</td>
<td>DDEML.DLL is not initialized.</td>
</tr>
<tr>
<td>16388</td>
<td>DMLERR_DLL_USAGE</td>
<td>General DDE library usage error.</td>
</tr>
<tr>
<td>16389</td>
<td>DMLERREXECACKTIMEOUT</td>
<td>Timeout waiting for an exec acknowledgment.</td>
</tr>
<tr>
<td>16390</td>
<td>DMLERRINVALIDPARAMETER</td>
<td>Invalid parameter to DDEML function call.</td>
</tr>
<tr>
<td>16391</td>
<td>DMLERR_LOW_MEMORY</td>
<td>Memory is becoming low.</td>
</tr>
<tr>
<td>16392</td>
<td>DMLERR_MEMORY_ERROR</td>
<td>Memory is exhausted.</td>
</tr>
<tr>
<td>16393</td>
<td>DMLERR_NOTPROCESSED</td>
<td>Receiving task was not interested in message.</td>
</tr>
<tr>
<td>16394</td>
<td>DMLERR_NO_CONV_ESTABLISHED</td>
<td>No DDE conversation could be established.</td>
</tr>
<tr>
<td>16395</td>
<td>DMLERR_POKEACKTIMEOUT</td>
<td>Timeout waiting for a poke acknowledge.</td>
</tr>
<tr>
<td>16396</td>
<td>DMLERR_POSTMSG_FAILED</td>
<td>Attempt to post a window message failed.</td>
</tr>
<tr>
<td>16397</td>
<td>DMLERR_REENTRANCY</td>
<td>The DDE library was re-entered during a blocking call.</td>
</tr>
<tr>
<td>16398</td>
<td>DMLERR_SERVER_DIED</td>
<td>DDE server has died.</td>
</tr>
<tr>
<td>16399</td>
<td>DMLERR_SYS_ERROR</td>
<td>A DDE call has caused a system error.</td>
</tr>
<tr>
<td>Number</td>
<td>Error String</td>
<td>Error Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>16400</td>
<td>DMLERR_UNADVACKTIMEOUT</td>
<td>Timeout waiting for an unadvised acknowledge.</td>
</tr>
<tr>
<td>16401</td>
<td>DMLERR_UNFOUND_QUEUE_ID</td>
<td>DDE queue id could not be found.</td>
</tr>
</tbody>
</table>
Appendix J. Third-Party Source Licenses

The following licenses are being used in this product:

License for Scintilla and SciTE

Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

License for ScintillaNET

ScintillaNET is based on the Scintilla component by Neil Hodgson.

ScintillaNET is released on this same license.

The ScintillaNET bindings are Copyright 2002-2006 by Garrett Serack <gserack@gmail.com>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

GARRETT SERACK AND ALL EMPLOYERS PAST AND PRESENT DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL GARRETT SERACK AND ALL EMPLOYERS PAST AND PRESENT BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
SHTTPD

Copyright (c) 2004-2005 Sergey Lyubka <valenok@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this soft-
ware and associated documentation files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy, modify, merge, publish, dis-
tribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IM-
PLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FIT-
NESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
Appendix K. Media Source Licenses

Some of the images, sound files, and videos distributed with this product are covered by one or more of the following licenses or terms of use:

- GNU Lesser General Public License (LGPL)
- Creative Commons Attribution 2.5 Generic license
- Creative Commons Attribution-Share Alike 3.0 Unported license
- Sound Jay Terms of Use
Appendix L. GNU General Public License

Copyright © 1989, 1991 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:
1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses,
in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that
users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose
of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type “show w”. This is free software, and you are welcome to redistribute it under certain conditions; type “show c” for details.

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something
other than “show w” and “show c”; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program “Gnomovision” (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.
Appendix M. GNU Lesser General Public License

This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.

Copyright © 1991, 1999 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Version 2.1, February 1999

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code; that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

We protect your rights with a two-step method:
1. we copyright the library, and
2. we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients
should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the Lesser General Public License because it does Less to protect the user’s freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users’ freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a “work based on the library” and a “work that uses the library”. The former contains code derived from the library, whereas the latter must be combined with the library in order to run.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called “this License”). Each licensee is addressed as “you”.

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

Section 1

You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.
b. You must cause the files modified to carry prominent notices stating that you changed
the files and the date of any change.

c. You must cause the whole of the work to be licensed at no charge to all third parties
under the terms of this License.

d. If a facility in the modified Library refers to a function or a table of data to be supplied
by an application program that uses the facility, other than as an argument passed
when the facility is invoked, then you must make a good faith effort to ensure that, in
the event an application does not supply such function or table, the facility still oper-
ates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is en-
tirely well-defined independent of the application. Therefore, Subsection 2d requires
that any application-supplied function or table used by this function must be option-
al: if the application does not supply it, the square root function must still compute
square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that
work are not derived from the Library, and can be reasonably considered independent
and separate works in themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you distribute the same
sections as part of a whole which is a work based on the Library, the distribution of the
whole must be on the terms of this License, whose permissions for other licensees extend
to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work writ-
ten entirely by you; rather, the intent is to exercise the right to control the distribution of
derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library
(or with a work based on the Library) on a volume of a storage or distribution medium
does not bring the other work under the scope of this License.

Section 3

You may opt to apply the terms of the ordinary GNU General Public License instead of this
License to a given copy of the Library. To do this, you must alter all the notices that refer
to this License, so that they refer to the ordinary GNU General Public License, version 2,
instead of to this License. (If a newer version than version 2 of the ordinary GNU General
Public License has appeared, then you can specify that version instead if you wish.) Do not
make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary
GNU General Public License applies to all subsequent copies and derivative works made
from that copy.

This option is useful when you wish to copy part of the code of the Library into a program
that is not a library.
Section 4

You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

Section 5

A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

Section 6

As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.
You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

Section 7

You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:
Section 8

You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 9

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

Section 10

Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

Section 11

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 12

If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 13

The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

Section 14

If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 15

BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PER-
FORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 16

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

###lt;one line to give the library’s name and a brief idea of what it does.###gt; Copyright (C) ###lt;year###gt; ###lt;name of author###gt;

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:
Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

###lt;signature of Ty Coon###gt;, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!
Cogent DataHub® WebView
Version 2.0
# Table of Contents

Introduction .................................................................................................................... 1  
Desktop WebView ........................................................................................................ 2  
Run Locally ............................................................................................................. 2  
Web Launch ............................................................................................................. 4  
DataHub Security .................................................................................................... 9  
Silverlight WebView ............................................................................................... 10  
System Requirements ............................................................................................. 10  
Configuration ......................................................................................................... 11  
Start the Editor ....................................................................................................... 14  
Working With DataHub WebView ............................................................................ 17  
Quick Start ............................................................................................................. 17  
Choose a Platform and Start WebView ................................................................... 17  
Add and Modify a Control ....................................................................................... 17  
Bind a Control to a Data Point ............................................................................... 18  
Save and View a Page ............................................................................................. 19  
Add a Symbol .......................................................................................................... 21  
Bind a Control to another Control ......................................................................... 22  
Set Symbol States ................................................................................................... 23  
User Access ............................................................................................................. 23  
Configure User Permissions .................................................................................... 24  
Log in Remotely ....................................................................................................... 26  
Pages .......................................................................................................................... 26  
Create, Open, Save, and Delete Pages .................................................................... 26  
DataHub and SkkyHub WebView Pages .................................................................. 26  
Page Size .................................................................................................................. 27  
The Grid .................................................................................................................... 27  
View and Zoom ........................................................................................................ 27  
Design and Run Modes ............................................................................................ 28  
Controls .................................................................................................................... 28  
Add, Copy, Resize, and Move Controls .................................................................... 28  
Grouping Controls .................................................................................................... 29  
Control Properties .................................................................................................. 30  
Common Properties ................................................................................................ 30  
Format Strings .......................................................................................................... 32  
Controls Listed by Category ..................................................................................... 34  
Property Binding ..................................................................................................... 36  
DataHub Point Binding ............................................................................................ 36  
Point Attribute Selection ......................................................................................... 38  
Simple Binding - Property Picker .......................................................................... 39  
Simple Binding - Copy and Paste ............................................................................ 41  
Managing Files ....................................................................................................... 42  
File Locations .......................................................................................................... 42  
Editing WebView XML Files .................................................................................... 44  
DataHub WebView Scripting ................................................................................... 45
Dynamic Binding ........................................................................................................... 46
  Dynamic Point Binding ........................................................................................ 46
  Combo Box control ................................................................................... 46
  List Box control ......................................................................................... 47
Dynamic Control and Symbol Binding ................................................................. 49
  Control Binding ......................................................................................... 49
  Symbol Binding ......................................................................................... 51
Creating a Template Page ................................................................................... 53
Customizing DataHub WebView .................................................................................... 56
Simple Branding .................................................................................................. 56
  Creating a custom login page .................................................................... 56
  Specifying text, icon, and URL targets ........................................................ 57
  Adding a favorite icon ............................................................................... 58
  Testing the results ..................................................................................... 59
Initialization Parameters ...................................................................................... 59
  Desktop WebView ..................................................................................... 59
  Silverlight WebView ................................................................................... 60
  Parameter List ........................................................................................... 62
Adding Controls .................................................................................................. 66
  Preparing the Visual Studio project ........................................................... 66
  The XAML file to reference the control ...................................................... 69
  The XML file for public properties and behavior of the control .............. 70
  Testing the results ..................................................................................... 72
Creating Custom Symbols ...................................................................................... 73
Creating Your Symbol Library .............................................................................. 73
  Create a symbol map ................................................................................ 73
  Create a symbol XAML file ........................................................................ 75
  Deploy your symbol set ............................................................................. 76
Symbol Animation .................................................................................................. 77
  Animation attributes ................................................................................. 77
  Animation types ......................................................................................... 78
  Conditional animation ............................................................................... 83
Scaling and Other Considerations ........................................................................ 83
  Scaling ....................................................................................................... 83
  Binding element properties to the symbol host control ......................... 84
  Limitations on XAML code within symbols ........................................... 84
  Compressing symbol files ......................................................................... 84
  Best Practices ............................................................................................ 84
Complete Example .............................................................................................. 85
WebView Controls ......................................................................................................... 92
  Advanced Check Box ................................................................................ 93
  Alarm List ................................................................................................. 94
  Boolean Converter ...................................................................................... 95
  Calendar ...................................................................................................... 96
  Circular Gauge 1 ........................................................................................ 97
  Circular Gauge 2 ........................................................................................ 98
Color Selector ................................................................................................................. 99
Color Selector .................................................................................................................. 100
ComboBox .......................................................................................................................... 101
Comparator ......................................................................................................................... 102
Condition Selector .............................................................................................................. 103
Control Panel ..................................................................................................................... 104
Date Picker .......................................................................................................................... 105
Filtered Data Table .............................................................................................................. 106
Hi/Low Indicator ................................................................................................................ 107
Horizontal Linear Gauge ................................................................................................. 108
Hyperlink Button ............................................................................................................... 109
Hyperlink Image .................................................................................................................. 110
Hyperlink Text ................................................................................................................... 111
Image .................................................................................................................................... 112
Left 90 Degree Gauge ........................................................................................................ 113
ListBox .................................................................................................................................. 114
MediaPlayer ....................................................................................................................... 115
One Input Calculator .......................................................................................................... 116
Point Data Table ................................................................................................................ 117
Polynomial Calculator ....................................................................................................... 118
Progress Bar ....................................................................................................................... 119
RadioButton ......................................................................................................................... 120
Range Mapper ..................................................................................................................... 121
Rising/Falling Indicator ....................................................................................................... 122
Semi-circular Gauge .......................................................................................................... 123
Series Chart ....................................................................................................................... 124
Shining Light ....................................................................................................................... 125
Simple Button ..................................................................................................................... 126
Simple Check Box ............................................................................................................... 127
Simple Ellipse ...................................................................................................................... 128
Simple Path ......................................................................................................................... 129
Simple Radial Gauge .......................................................................................................... 130
Simple Rectangle ............................................................................................................... 131
Slider .................................................................................................................................... 132
Symbol ................................................................................................................................. 133
System Information .............................................................................................................. 134
Text Entry Field .................................................................................................................. 135
Text Label ........................................................................................................................... 136
Thermometer ....................................................................................................................... 137
Three Indicator Radial Gauge ........................................................................................... 138
Three Point Slider .............................................................................................................. 139
Time Picker .......................................................................................................................... 140
Timer ..................................................................................................................................... 141
ToggleButton ...................................................................................................................... 142
Top Sweep Gauge .............................................................................................................. 143
Trend .................................................................................................................................... 144
Two Input Calculator ................................................................. 145
Vertical Linear Gauge ............................................................... 146
Introduction

DataHub WebView is a state-of-the-art, rich internet application for designing and delivering high-quality, real-time displays in Windows Desktop or Internet Explorer. All page creation and editing is done using a built-in editor, and page updates can be automatically published as soon as changes are saved. Page designers have access to standard controls, gauges, and 4,000 industry-standard symbols, each of which can be easily configured to recognize condition states and to graphically notify operators of process status and anomalies. All controls feature powerful, "anything-to Anything" data binding.

Advantages of DataHub WebView

- **Edit screens anywhere** Just open your web browser, type the address of your DataHub Web Server, log in to the system and, based on your access permissions, you can build, edit and view WebView screens from anywhere with network/Internet access.
- **No coding required** Build all your screens using the powerful built-in graphical environment. Complete point and click freedom, with no code in sight.
- **No development system required** You do not need to install a development system on your computer in order to build WebView screens. The development interface is provided to you, in your browser, by the web server. No compiler necessary.
- **No deployment required** Because the screens you build in WebView are saved on the web server, you never have to deploy your changes to other users. When you are editing a page and you want to show your colleagues, you simply save your changes, and tell your team to reload the page in their browser. No deployment, no complicated file updates.
- **Collaborate on development** With no limit to the number of users accessing the system, you can have a team of developers building pages at the same time. Depending on the permissions you give to your team members you can have them edit everyone's pages, or restrict access so they can only edit the pages they build.
- **Build entire multi-page HMI applications** with hyperlinks between pages, just like a
web site or standard HMI system.

- **Security at every level** The permissions based security model allows you to define very precise privileges to each user. You can define certain users to have read only access, while other can view and make changes but not access the development interface. In WebView, the security model was one of the fundamental primary requirements, not an afterthought.

- **Specify sophisticated graphical interactions between controls** using WebView's powerful property binding feature. Property binding allows you to associate the input values for one control, with the properties of other controls. Simpler to do than to describe, this saves you time and adds significant power to your WebView screen.

- ** Comes with a standard set of built-in controls** such as trends, gauges and sliders. In addition, WebView also ships with a complete set of Symbol Factory™ symbols, which offer thousands of industry standard symbols for a variety of industries. WebView is able to provide these and other future third party symbols.

**Desktop WebView**

The original version of DataHub WebView was implemented using Microsoft Silverlight, which allows anyone to create, save, and load pages in Internet Explorer. Because support for Silverlight ends in 2021, we now provide a Windows desktop version of WebView called Desktop WebView.

Desktop WebView looks and behaves very much like the Silverlight version, including a facility for web launch, and the ability to store and save user content centrally on the DataHub computer. Users of the original version of WebView should find Desktop WebView just as convenient and easy to use. Both versions ship with DataHub Version 9, and any user content that is developed in either the desktop or Silverlight versions of WebView will be immediately available in the other with no additional effort.

**Run Locally**

1. To run Desktop WebView on your local computer, from the Start menu go to the Cogent programs and select the Desktop WebView program.

   ![WebView](image)

   This will open the Desktop WebView log-in screen:

---

1Symbol Factory™ is a trademark of Reichard Software Corporation. Symbol Factory graphics are included with WebView under license from Software Toolbox and Reichard Software Corporation.
2. To log in, enter **Local, admin, admin** for **Organization**, **Username** and **Password**, respectively. Typically the **Advanced Options** are not needed on a local connection. Logging in will open the Desktop WebView interface:
You are now ready to start creating pages.

Web Launch

The DataHub installs a local copy of Desktop WebView, which you can run as described in the previous section. You can also web launch Desktop WebView from a different computer that does not have the DataHub installed, using a Desktop WebView app. The DataHub provides a special program, the Skkynet Web Application Manager, that helps you securely manage this process. Here's how you download the Web Application Manager from a remote system, and use it to web launch the Desktop WebView app:

1. Ensure that the DataHub instance you want to access is running, and that its Web Server feature is enabled.
2. Open a web browser and type in the IP address or network name for the DataHub computer. If the Web Server feature on that DataHub is configured to use a port other than the default port 80, include that port number in the network address. For example, if the remote DataHub is at 192.168.3.12 and its Web Server is configured to use port 8080, then the network address would be 192.168.3.12:8080. This opens the Web Launch Page.

3. Click the Download Web Application Manager button and follow the instructions.
to download and install the Skkynet Web Application Manager. Now you are ready to launch an application.

This installation is signed with a valid code signing certificate from Skkynet. If Windows cannot validate the certificate, do not install the software.

Launching the Desktop WebView app

Once you have installed the Web Application Manager, follow these steps:

1. From the Web Launch Page, click on the Launch Desktop WebView button. A dialog will ask if you want to switch to the Web Application Manager.

2. Click Accept. The Web Application Manager will open, populated with the parameters required for you to connect to the remote DataHub system and download the Desktop WebView app.

3. Click the green Run button.

The files needed to run the Desktop WebView app will be downloaded, and a dialog window will display information about the program files, with details of the security certificate used to sign them.

Please see SSL Certificates for more information about SSL certificates in Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
4. Check the signing certificate and affected files to determine whether or not to accept them, based on the following criteria:

   • If the Signing Certificate region of this dialog is not green, then the Status will be Invalid. That indicates that the certificate cannot be verified. Do not click Accept unless you know why you are seeing a rejected certificate.

   • If the Signing Certificate is from an organization other than Skkynet Cloud Systems Inc. then it indicates that another company's software has been installed on the server and is being delivered as part of the application. If you do not believe this to be legitimate, do not click Accept.

   • If you click Accept it means that you trust the app source files. They will be saved on your computer and the app will run. Your choice will be remembered, and the next time you web launch this app you will not have to accept the files again.

      If, for some reason you want to revoke your acceptance of the downloaded files, you can select the profile in the Web Applications Manager and click the Clear Authorization button.

   • If you click Reject the files required to run the app will be deleted. You will also see a list of files you just rejected and be given the chance to change your mind. To revoke your choice:

      1. Click the Reset DLL Authorization button.
      2. Click OK to close the dialog window.
      3. Download the files again by clicking the green Run button in the Web Application Manager.

5. As soon as the app files have been accepted, Desktop WebView will run:
6. To log in, enter **Local, admin, admin** for **Organization, Username** and **Password**, respectively. (Please see **DataHub Security** for details about changing the user name and password.)

**Advanced Options**

The following advanced options are available.

**Host Name**

The name or IP address of the computer on which the DataHub is running.

**Web Port**

The HTTP port number on which the DataHub Web Server is listening. The default is 80.

**Use HTTPS**

Enables the HTTPS protocol, and changes the **Web Port** default to 443.
**Data Port**

The number of the port used to connect to the DataHub data feed. The default is 4502.

**Use SSL**

Enables SSL, and changes the Data Port default to 4503, or 443 when combined with **Use WebSocket**.

**Use WebSocket**

Enables the WebSocket protocol, and changes the Data Port default to 80, or 443 when combined with **Use SSL**.

**Accept Untrusted Certificates**

Ignores security certificate warnings.

7. The first time the Desktop WebView app is used, it will download a set of WebView controls and support files which enable you to design and build WebView pages. These files are also certificate signed, so you will see a second list of files you need to accept or reject, as above. If you reject the files, you will not be able to place or see controls in your WebView screens.

WebView allows you to create and deliver custom controls to your users. These custom controls are delivered to the client as DLLs that contain executable code. If you have created a custom control, or are using somebody else’s custom control, it must be signed with a code signing certificate. If the certificate is not valid the dialog background will be red, and the certificate status will be Invalid. You may still accept the controls if you know they are from a trusted source. Never accept controls that are delivered from the Internet, including from Skkynet, that are not signed with a valid certificate.

Click **Accept** to download the necessary files and continue. Your choice will be remembered, so you will not see this again the next time you run the app.

It is highly unlikely, but if for some reason you want to revoke your acceptance of the downloaded WebView controls and support files, then you will need to navigate to the following location and delete the settings.xml file.

C: \Users\user name\AppData\Roaming\Skkynet\Cache\your IP address_80\ControlAssemblies

You are now ready to **start creating pages**.
Create a Shortcut

Here are two ways to make a shortcut to quickly start Desktop WebView.

1. While Desktop WebView is running, you can pin the program to your task bar, creating a short cut.

2. From the Web Launch Page you can drag the direct link: `nnn.nnn.nnn.nnn.web-view`, onto your desktop.

With either of these shortcuts you can start the program with one click; no need to visit the Web Launch Page.

DataHub Security

Once you have tested Desktop WebView with the default username and password (`admin`, `admin`), you may want to change the `admin` password or add other users with different access permissions. Here is how you can do that.

1. Start the DataHub, right click on the Cogent DataHub system-tray icon and choose `Properties`.

2. In the Properties window, select `Security`.
3. Click the **Configure Permissions** button to open the Edit Permissions window.
4. Click on the **Users** tab and create a new user, for example **WebViewUser**.

5. You can now assign access permissions for this user. To give permission to access all WebView features as well as read/write permissions on data, check the **BasicConnectivity** and **WebView** boxes. You can also create custom groups with more restricted permissions, and assign users to them. Please refer to **Security** for more information.

6. When you are finished, click the **Apply and Close** button.

### Silverlight WebView

The system requirements and DataHub configuration needed for running WebView on Internet Explorer are as follows:

#### System Requirements

- **Windows** X86 or x64 (64-bit mode support for IE only) 1.6 GHz or higher processor with 512-MB of RAM.

- **Internet browser for WebView** Internet Explorer 9 or later.

#### Minimum and Recommended

- **Hard disk** minimum 70 MB, recommended 70 MB plus any additional space for historical files, log files. The Cogent DataHub does not use substantial space beyond the minimum unless you configure it to use more.

- **Memory** minimum 50 MB, recommended 500 MB. The DataHub will consume more memory with more data points configured, and with more client connections. We recom-
mend having enough spare memory to load a large data set. Swapping will reduce performance.

**Number of processor cores** minimum 1, recommended 2. An extra core will allow busy connections and scripts to run on a separate core and will help to keep the GUI responsive. The DataHub can use as many cores as you provide.

**Network protocol between WebView and DataHub** WebView needs access via TCP/IP on three ports:
- The **web server port** can be any port number (default is 80).
- The **Silverlight policy server port** must be port 943.
- The DataHub **plain-text tunnel port** port must be in the range of 4502-4534 (default is 4502).

UDP is not used.

**Configuration**

**DataHub Configuration**

Certain parameters of WebView can be configured from within the Cogent DataHub Properties window:

![WebView Configuration (Silverlight)](image)

**Data Domains Visible to DataHub WebView**

Check the data domains that you want to access from WebView. Use the **Add...** button to create and add new domains.

**Start in Run mode**

Allows you to start in Run mode, rather than Design mode, with these options:
Kiosk mode

Presents just the working screen of the web browser, with no border, menus, URL entry field, etc. To escape from Kiosk mode (and close the browser), press **Alt + F4**.

Disable Design mode

Prohibits any switch from Run mode to Design mode, whether running in Kiosk mode or normally.

Disable data writes from client

Prevents the web client from accessing DataHub point values.

Show page information icon

Shows or hides the page information icon.

Load a page at startup

Allows you to specify a page that will automatically load when WebView starts.

Use a custom branding folder

Allows you to specify a folder for holding custom branding information. For details, please refer to Customizing DataHub WebView.

Launch WebView in a browser

Provides a convenient way to start WebView to check this configuration.

DataHub WebView requires the DataHub to be configured as a tunnelling master. Please refer to Tunnel/Mirror Master in the DataHub manual for details.

Internet Explorer Configuration

For best results, anyone editing or viewing WebView pages in Internet Explorer should modify its settings as follows:

1. Open the Internet Explorer menu option **Tools / Internet Options** and select the Browsing History Settings button:
2. Change the cache update algorithm to check for new versions **Every time I visit the webpage**:

Do *not* select the option **Automatically**. This will cause Internet Explorer to try to guess whether to check for updates to its cache based on a heuristic algorithm that attempts to guess how frequently it should check. This algorithm will usually not check for updates, and the behaviour described below will occur.

**Explanation**

Internet Explorer maintains a local cache of pages that it has visited recently, and when you make a request for a page it will serve the page from the cache instead of from the web server. This can speed up browsing, but it makes the behaviour of WebView unpredictable.

Normally a web server can mark a page as "no-cache" in which case the browser will always re-load the page on each visit. That is not what we want in the case of WebView. We want the pages to be re-loaded only if they have changed on the server. The HTTP proto-
col allows for this, and this is the best method for ensuring that the page you are looking at is up to date. Essentially, the browser makes a request from the server, telling the server the timestamp of the cached version of the page. If the cached version is up to date, the web server just returns an indication that the browser should use its cached copy of the page. This is efficient since the amount of information transmitted is small.

Unfortunately, Internet Explorer does not default to the best behaviour. Instead, it contains a heuristic algorithm that uses the cached copy of a page without ever consulting the web server. This can result in the following condition in WebView:

1. You load a WebView page
2. You edit the page and then save it. The page is correctly stored on the DataHub Web Server.
3. You re-load the page you have just edited. Internet Explorer retrieves the page from its cache without consulting the DataHub Web Server.
4. WebView receives the old page from Internet Explorer's cache and your changes appear to have vanished. At this point, the copy of the page stored in the DataHub Web Server is correct, but Internet Explorer never asks for it. To you, it appears as if the page was not saved and your edits were lost.

**Start the Editor**

1. In the Cogent DataHub Properties window, select WebView.
2. In the DataHub WebView configuration window, select the Data Domains that you wish to access data from, and press the Apply button.

See also the section called “Configuration” for more details about configuration.

3. Once configured, there are two ways to start WebView:
A. From here in the Properties window, press the **Launch DataHub WebView in a browser** button.

B. From a web browser:

1. Open a web browser like Internet Explorer, Firefox, or Chrome.
2. Type in the DataHub WebView default URL:
   
   ```
   http://localhost/Silverlight/DataHubWebView.asp
   ```

   Using `localhost` opens WebView on your computer. If you need to connect to WebView on another computer, instead of `localhost`, use that computer's IP address in the URL, and keep everything else the same.

Either of these will open the WebView application in your web browser.

4. Enter the password **admin** to start the WebView Editor. See also the section called “Configure User Permissions” for more details about security, user names, and passwords.
Introduction
Working With DataHub WebView

Quick Start

Here is a brief example of how to start WebView, add a control to a page, edit a property and animate the control with live data, and then save and view the resulting page.

Choose a Platform and Start WebView

WebView runs on two platforms: Windows Desktop and Internet Explorer. Please refer to the section called “Desktop WebView” or the section called “Silverlight WebView” for details about these options. According to the platform you choose, you have three options for starting WebView:

1. Run Desktop WebView locally.
2. Web launch Desktop WebView.
3. Run Silverlight WebView in Internet Explorer.

Once WebView is started, you are ready to begin building pages.

Add and Modify a Control

1. Find the Circular Gauge 2 control button among the controls in the editor, and click it.

A copy of the Circular Gauge 2 control will appear in the blank page.

2. Now let’s adjust a property of the gauge. In the Properties list on the left, find the Scale Properties, expand the list, and in the Scale Label Font Size, enter 20.
Press Enter, and the font size of the numbers on the gauge will expand to 20 points.

See also the section called “Control Properties” for more details about working with control properties.

**Bind a Control to a Data Point**

Many of the controls in WebView can be bound to certain variables, so that whenever the variable changes, the value or appearance of the control changes as well. For example, a gauge or meter can be bound to a DataHub point to display changes to the point in real time. In this example we bind the gauge we created above to display the value of the DataPid point Pv.

1. Start the DataPid program that is included in your Cogent DataHub archive to generate some test data.
2. Open the Basic Properties of your gauge, and click the arrow button on the right side of the Current Value row.

A Binding selection box will appear.

3. In the Binding selection box, click the down arrow to open the list, and select Point.
This will activate the point selection entry field.

4. We want to connect to the DataPid point DataPid:PID1:Pv, so enter just Pv. All the points that have "Pv" in their names will appear.

Select the point DataPid:PID1:Pv. Once selected, the data will start updating in the value entry, and the gauge needle will start to move.

See also the section called “Controls” for more details about using controls, or the section called “Property Binding” for binding data them to DataHub points.

Save and View a Page

1. To save the page, you can click the Save button , or choose Save from the File menu, or press Ctrl + Shift + S.
Specify a filename for the page.

Any page created in WebView version 1.4 and earlier (that was distributed with DataHub version 7.x) gets upgraded to the current version of WebView as soon as it is opened. If you save this page, the conversion is permanent, and you will be unable to load the page in the older version of WebView. If you expect to revert to the older version, you should create a backup of your page before saving it in the latest version of the DataHub.

2. To enter Run mode and view your page, click on the Enter Run Mode button or press Ctrl + Shift + R.

Your page will appear in the web browser as a user would see it, with all the controls fully animated and functional.
To exit Run mode and return to Design mode, click on the **Exit Run Mode** button or press **Ctrl + Shift + R**.

See also the section called “Pages” for more details about saving and viewing pages.

**Add a Symbol**

1. Find the Symbol control button at the bottom of the editor, and click it.

A copy of the Symbol control will appear in the page. This one control can be used to represent any symbol in the symbol library, which contains thousands of different symbols.

2. In the Properties list, for the **Symbol Set**, choose **Symbol Factory**. For **Category** choose **Pumps**, and for **Symbol**, choose **Cool pump**.
The generic symbol icon should change into a symbol of a pump.

**Bind a Control to another Control**

Most controls can bind their properties to other controls, so that when the first control is modified, the bound control gets modified automatically. Here's an example, binding the value of the pump we just created to the value of the gauge.

1. Click the gauge, and in the **Basic** properties, right-click the **Current Value** row, and select **Copy Reference**.

2. Click the pump, and in the **Input** properties, right-click the **Input Value** row, and select **Paste Link**.

The pump will take the same values as the gauge, and turn green, the default non-zero color for this symbol.
Notice that the **Current State** is **State 1**, for **True**. This default can be changed, as explained below.

See also the section called “Property Binding” for more details about binding control properties.

**Set Symbol States**

Most symbols can be set as booleans, to show on/off states, and many can also display multiple states. Here we’ll change the default boolean settings and colors of the pump to display three different states.

1. Click on the pump, and uncheck the **Treat as Boolean** box.

2. In the **State 0** properties, for **Value Is Less Than or Equal** enter a value of 35. Then change the **Color** to **PowderBlue**.

3. In the **State 1** properties, enter a value of 65 and change the **Color** to **MediumBlue**.

4. In the **State 2** properties, enter a value of 100 and change the **Color** to **Navy**.

Now, whenever the gauge value is between 0 and 35, the pump color will be light blue, 36 to 65 medium blue, and 66 to 100 dark blue. Note that for each state, you enter the maximum value, while the minimum value is controlled by the previous state.

**User Access**
Configure User Permissions

User permissions and passwords are assigned according to groups in the Security option of the DataHub Properties window. Here is an example of how to configure two different groups: operators and page designers, and then assign users to them.

1. Open the Security option of the Properties window and click the **Configure Permissions** button.

2. Click on the **Groups** tab. This will allow you to configure a group of users that will all inherit the same permissions.

3. Create a new group by typing in the name: **Operator**.
4. Select the following permissions:
   • In the **Connections** section, to allow a connection, select **Connect**.
   • In the **Data** section, to give access to real-time data, select **Read** and **Write**.
   • In the **HTTP** section, select **Connect** to allow DataHub WebView to connect to the DataHub Web Server.
   • In the **Web** section, to allow access to web pages, select: **ViewPage** and **ViewOtherOwnerPage**.
   • And in the **DataHub WebView** section, to allow basic interaction with the WebView application, select: **Connect**, **BrowseInternet**, and **ViewOnlineHelp**.

5. Click on the Users tab.

![User Management Interface](image)

6. Add a user name, with no spaces in it, for example, **JohnDoe**, and enter a password for him.

   The password is not stored in the system. If a user forgets his password, you cannot retrieve it. You will need to assign him a new password.

7. Check the **Operator** box to make **JohnDoe** a member of the Operator group. Notice that all of the group's permissions are given to **JohnDoe**.

8. Another useful group to create would be Designers, who have permission to design pages. These users would need the same **Connection** and **HTML** permissions as above but could take more, even all, of the **DataHub WebView** options, as needed.

9. Once that group is created, you can assign designer users to the group.

10. When finished, press the **Apply** button to write your configuration to the DataHub.
permissions database.

Now, when you launch WebView and log in as JohnDoe, you should go straight to the Start page in Run mode, and have no access to Design mode. If you log in as a designer, you will be placed in Design mode.

Log in Remotely

Any user can very easily log in to DataHub WebView remotely from another computer on the network, or over the Internet. Just open your web browser and navigate to this URL:

http://IP_or_computer_name/Silverlight/DataHubWebView.asp

Where IP_or_computer_name is the network IP address or DNS computer name of the computer running the Cogent DataHub.

Pages

If you are using Internet Explorer, please ensure that it is configured properly for use with WebView.

Create, Open, Save, and Delete Pages

To create a page click on the Pages tab and then click on the New button in the Toolbar. You can also create a page from the Edit menu, or by pressing Ctrl + Shift + N.

To open a page click on the Pages tab and then click on the Open button next to the name of the page. You can also open a page by double-clicking on the name of the page in the Pages tab.

To save a page simply click on the Save button. You can also save a page from the Edit menu, or by pressing Ctrl + Shift + S.

Any page created in WebView version 1.4 and earlier (that was distributed with DataHub version 7.x) gets upgraded to the current version of WebView as soon as it is opened. If you save this page, the conversion is permanent, and you will be unable to load the page in the older version of WebView. If you expect to revert to the older version, you should create a backup of your page before saving it in the latest version of the DataHub.

To remove a page from the Pages tab, you will need to manually delete the page from the DataHub WebView installation directory. Please see File Locations for more information.

DataHub and SkkyHub WebView Pages

WebView is available on two platforms: DataHub and SkkyHub. Prior to DataHub v8.0, WebView had its own version numbering, with the most recent version number being 1.4.
With the introduction of DataHub v8.0, the WebView version number has been changed to 8.0, and will always correspond to the latest DataHub version.

Compatibility between DataHub WebView pages and SkkyHub WebView pages is as follows:

1. DataHub WebView 8.0+ and SkkyHub WebView pages are compatible. Pages that you edit and save on one platform can be moved to the other platform, and be edited and saved, and vice-versa.
2. DataHub WebView 1.4 (DataHub 7.x) and SkkyHub WebView pages are not compatible. Each page may be edited only on the platform on which it was first created.
3. Using DataHub WebView version 8.0+ to open, edit, and save a DataHub WebView version 1.4 page (i.e., one that was previously saved with DataHub version 7.x) automatically and irreversibly “upgrades” the page to WebView 8.0+ when the page gets saved.
4. If you want to use a local version of DataHub WebView to modify pages intended to be deployed to SkkyHub WebView, you must upgrade to Cogent DataHub version 8.0+

Page Size

To change page size enter a number of pixels for the width and height of the W: and H: entry fields at the bottom right corner of the editing window.

![Page Size](image)

The Grid

To show gridlines click the Show gridlines button, or type True in the Tools menu, Options dialog, Design Mode list, Show gridlines entry.

To change the grid size enter a number of pixels in the Grid entry field at the bottom right corner of the editing window.

![Grid Size](image)

To snap controls to the grid click the Snap to grid button, or type True in the Tools menu, Options dialog, Design Mode list, Snap to grid entry.

View and Zoom

To view the page at a specific size, click the Page Zoom button, which opens a list of zoom levels, and choose the level you need. There are several other ways to zoom in and out, to make resizing the page convenient.
To fit the page into the window, click the Fit button, use the View/Zoom menu, or press Ctrl + Shift + Z.

To zoom in and out, click the Zoom In or Zoom Out buttons, use the View menu, or press Ctrl + Shift and spin the mouse wheel up or down.

To focus your zoom on a specific location in the page, click the Set Zoom Focal Point button, or click that option in the View/Zoom menu. Then click the page where you want to focus your zoom.

To zoom on a control, click the control, and then from the View/Zoom menu check the Zoom on Selected Control option.

Design and Run Modes

At log in DataHub WebView checks for user and editing permissions. If you log in with a user name that does not have editing permissions, then WebView will automatically open in Run mode and you will not be able to switch to Design mode. If you do have permissions to edit and create new pages, then WebView will open in Design mode.

To enter Run mode from Design mode, click on the Enter Run Mode button or press Ctrl + Shift + R.

To exit Run mode and return to Design mode, click on the Exit Run Mode button or press Ctrl + Shift + R.

To display the Kiosk View which removes the toolbars in Run mode, go to Design mode, and from the Edit menu, select Run Mode Options and check Use Kiosk View.

If you have configured Run mode to display in Kiosk View, there will be no toolbar at the top of the page, so you will need to use Ctrl + Shift + R to exit Run mode.

Controls

Add, Copy, Resize, and Move Controls

To add a new control to your page, you have two options:
1. In the Controls tab, find the control in its appropriate group and click the Add button.
2. Click that control's button in the Control Toolbar at the bottom of the page to put the control in the center of the page.

Pressing the Shift key while you click on the control button lets you manually position the control on the page. While positioning the control, pressing Ctrl + Shift and turning the mouse wheel allows you to zoom in and out.

To copy a control select it, and from the Edit menu choose Copy, or press Ctrl + Shift + C.

To paste a control select it, and from the Edit menu choose Paste, or press Ctrl + Shift + V.

To resize a control select it and resize it with one of the black resize handles. Or, enter a width and height in the W: and H: entry fields at the bottom right corner of the editing window.

To move a control select it, and move it with the mouse. For precise movements, you can show gridlines, and snap controls to the grid. Or, enter X and Y coordinates (distance from the top-left corner) in the X: and Y: entry fields at the bottom right corner of the editing window. Alternatively, you can use the cursor keys as follows:

• move by 1 pixel: Ctrl + arrow keys
• move by 10 pixels: arrow keys
• move by 100 pixels: Ctrl + Shift + arrow keys

Grouping Controls

Controls can be grouped together, forming essentially a single large control.

To group controls click the controls that you want to be in the group, and then click the Group button  

This is also available from the right-click pop-up menu.
To ungroup controls  click the group that you want to ungroup, and then do choose one of the following:

- Click the Ungroup button ᵂ to preserve the size and position changes you made while the controls were grouped. This is also available from the right-click pop-up menu.

- Click the Cancel Group button ᵆ to discard all changes that you made to the group. This is also available from the right-click pop-up menu.

To access and change the properties of any control in a group

1. Hover your mouse over the control you want to select.
2. Tap the Ctrl key on your keyboard.
3. Click the control in the group that you need to access.

If you have a group within a group, you can access a single cell like this:

1. Hover your mouse over the outer group.
2. Tap the Ctrl key on your keyboard.
3. Hover your mouse over the inner group that you need to access.
4. Tap the Ctrl key twice on your keyboard.
5. Click the control that you need to access.

Control Properties

Each control has a number of properties associated with it. There are common properties (see below) shared with each control, as well as other properties unique to that particular control. When you click on a control, the Properties tab opens, listing the properties by groups, with the Basic Properties group viewable. The Basic Properties are the ones that you'll probably use most often. Below these are groups of other properties unique to the control, followed by the common properties.

To change the value of a property  click the control and type in or select the value.

To bind the properties of a control  to a DataHub point or another control, please refer to the section called “Property Binding”.

Common Properties

The common properties shared by all controls include:

Background, Border and Margin

Background

A color for the background of the control.

Border

A color for the border of the background of the control.
Border Thickness
The thickness of the background border, in pixels, for the left, top, right, and bottom borders, respectively. Adding to this thickness will reduce the visible size of the control.

Border Corner Radius
The radius of each corner of the background border, in pixels, for the top-left, top-right, bottom-right, and bottom-left corners, respectively.

Content Margin
The width of the margins, in pixels, for the left, top, right, and bottom borders, respectively. Adding to this width will reduce the visible size of the control.

Background Image

Image File
A file to use for a background. To add images to the DataHub WebView library, you can copy your image files to the following directory:

```
[OrgLocal]\Images
```
If you copy images to a subdirectory, then they will appear together in the image file selector within the editor.

Image Width
The width of the image, in pixels.

Image Height
The height of the image, in pixels.

Image Alignment
Aligns the background image with a corner, side, or middle of the control, or stretches it to fill the whole area.

Image Opacity
A number between 0 (transparent) and 1 (fully opaque).

Image Margin
The width of the margins, in pixels, for the left, top, right, and bottom sides of the image, respectively. Adding to this width will reduce the visible size of the image.

Image Rotation (degrees)
A number of degrees to rotate the image to the right.

Flip X-Axis
Flip the image top-to-bottom.

Flip Y-Axis
Flip the image right-to-left.

Content Visibility and Appearance

Visible in Run Mode
Will this control be visible in Run mode?
Content Opacity
A number between 0 (transparent) and 1 (fully opaque).

Static Rotation (degrees)
A number of degrees to rotate the control to the right.

Maintain Uniform Size for Static Rotation
Will the control change size to fit its container when rotated?

Clip Content
Not yet documented.

Flip X-Axis
Flip the content of the control top-to-bottom.

Flip Y-Axis
Flip the content of the control left-to-right.

Position and Size
Left
A number of pixels specifying the distance of the top left corner of the control from the left side of the page.

Top
A number of pixels specifying the distance of the top left corner of the control from the top of the page.

Width
A number of pixels specifying the width of the control.

Height
A number of pixels specifying the height of the control.

Content Animation
Is Content Rotating
Specifies if the control content rotates.

Animated Rotation (rpm)
Specifies the speed, in rotations per minute, of the control content.

Format Strings
Controls like text labels, gauges, and level indicators that can display text or numbers have a Format String property that allows you to format the text. This is particularly useful for numbers.

Standard Numeric
Standard numeric values use Windows Standard Numeric Format Strings.
Examples:

<table>
<thead>
<tr>
<th>Input Value</th>
<th>Format String</th>
<th>Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1234</td>
<td>E2</td>
<td>1.23E-001</td>
</tr>
<tr>
<td>0.1234</td>
<td>P2</td>
<td>12.34%</td>
</tr>
</tbody>
</table>

The following formats are supported:

**Numeric type** | **Format string**
---|---
Decimal | D or d
Exponential | E or e
Fixed | F or f point
General | G or g
Number | N or n
Percent | P or p
Round-trip | R or r
Hexadecimal | X or x
All digits | @

Applying format strings may cause precision to be lost if there is a long string of digits in the number. To preserve the entire string of digits, set the format string to @. This is specific to WebView, and not a Windows format string.

**Custom Numeric**

Custom numeric values use Windows Custom Numeric Format Strings, which typically consist of one or more instances of the custom numeric specifiers 0 and #. Two- and three-part custom numeric format strings are also valid, using semi-colons to delimit the parts.

Examples:

<table>
<thead>
<tr>
<th>Input Value</th>
<th>Format String</th>
<th>Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>5678</td>
<td>#,##0.00</td>
<td>5,678.00</td>
</tr>
<tr>
<td>5678</td>
<td>My 0 data points.</td>
<td>My 5678 data points.</td>
</tr>
<tr>
<td>123.456</td>
<td>#.#;(#####)</td>
<td>123.5</td>
</tr>
<tr>
<td>-123.456</td>
<td>#.#;(#####)</td>
<td>(123)</td>
</tr>
<tr>
<td>0</td>
<td>#.#;(#####);zero</td>
<td>zero</td>
</tr>
</tbody>
</table>

**Text Values**

You can build composite strings by using the `{0}` placeholder to insert a text value.
Examples:

<table>
<thead>
<tr>
<th>Input Value</th>
<th>Format String</th>
<th>Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>The alarm is {0}</td>
<td>The alarm is OFF</td>
</tr>
<tr>
<td>High</td>
<td>Tank level: {0}</td>
<td>Tank level: High</td>
</tr>
</tbody>
</table>

**Dates**

Both standard and custom date format strings are supported, using Windows Standard or Custom Date and Time Format Strings, respectively.

Examples:

<table>
<thead>
<tr>
<th>Input Value</th>
<th>Format String</th>
<th>Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 9, 2016, 3:58 PM</td>
<td>d</td>
<td>05/09/2016</td>
</tr>
<tr>
<td>May 9, 2016, 3:58 PM</td>
<td>f</td>
<td>Tuesday, 09 May, 2016 15:58</td>
</tr>
<tr>
<td>07/14/2016 08:15:25</td>
<td>yyyy-MM-dd hh:mm tt</td>
<td>2016-07-14 08:15 AM</td>
</tr>
</tbody>
</table>

**Controls Listed by Category**

WebView controls are arranged in the following categories. You can click on the control icon for more information about the specific control.

- **Alarms**

- **Charts**

- **Common Input Controls**

- **Configuration**

- **DateTime**
• DTC: Palettes

• DTC: Program Blocks

• Gauges

• Media Controls

• Navigation

• Notification

• Shapes

• Symbols

• Text Controls
Property Binding

DataHub WebView provides extensive support for property binding, allowing the properties of one control to be bound to other controls, or to DataHub point values. These three binding options are available on many control properties:

- **None** allows you to enter a static value, not bound to anything.
- **Point** lets you bind a control property to the value of any DataHub point. When the point changes value, the property changes value with it. For example, you can bind the current value of a gauge to a DataHub point, animating the gauge indicator with live data.
- **Simple** lets you bind a property of a reference control to the property of a linked control. Whenever you change the value of the property on the reference control, the property will change on any linked controls. For example, if you bind the color and size of several buttons to a single, reference button, whenever you change the color or size of the reference button, all the other buttons will change too. Any control with bindable properties can be used as a reference control, a linked control, or both.

There are several features of property binding that facilitate page design, such as:

- **Simple bindings are universal** so that any control can be bound to any other control, as long as the bindings are compatible (eg. value-to-value, color-to-color, etc.).
- **The Property Picker shows compatible properties** for simple bindings.
- **For simple bindings, properties can be both references and links** which means that bindings can be chained, allowing controls to act simultaneously as a references to other controls or be linked to them, in any combination.
- **Simple and Point bindings can be combined** in a single control, which allows you to get your data from the DataHub, and the control appearance from another control.
- **Copying a control** will copy all property bindings. So, if you need a number of similar controls to share certain properties, you can make a reference control and one linked control, and then copy the linked control as many times as needed.

**DataHub Point Binding**

To create a **Point** binding you can follow this example, where we bind the indicator value of a gauge to a DataPid point:
1. Start the DataPid program that is included in your Cogent DataHub archive to generate some test data.
2. Open a blank page and add a Circular Gauge 2 control.

3. Open the Basic Properties of the gauge, and click the binding button on the right side of the Current Value row.

   A Binding selection box will appear.

4. In the Binding selection box, click the down arrow to open the list, and select Point.

   This will activate the point selection entry field.

5. To connect to the DataPid point DataPid:PID1:Pv, you can enter just Pv. All the points that have "Pv" in their names will appear.
Select the point **DataPid:PID1:Pv**. Once selected, the data will start updating in the value entry, and the gauge needle will start to move.

### Point Attribute Selection

When binding a point, WebView provides a choice of point attributes that are available for the control you are working with. These attributes may include, in various formats, the point name, the quality of the connection, the timestamp of the most recent value change, and the value of the point itself. These are chosen through right-clicking on the small arrow to the right of the text-entry field, and selecting from the drop-down menu that appears:

Here are the possible choices, availability depends on the control you are working with:

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>Point Attribute</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>n.ai</td>
<td>The full point name, no domain.</td>
<td>PID1.Pv</td>
</tr>
<tr>
<td>ab</td>
<td>The abbreviated point name.</td>
<td>Pv</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Point Attribute</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td>dn.ab</td>
<td>The full point name, with domain name.</td>
<td>DataPid:PID1.Pv</td>
</tr>
<tr>
<td>Qt</td>
<td>The point quality, in plain text.</td>
<td>Good</td>
</tr>
<tr>
<td>Qs</td>
<td>The code for the point quality.</td>
<td>192</td>
</tr>
<tr>
<td>s</td>
<td>The full date and time.</td>
<td>09/02/2011 13:09:15</td>
</tr>
<tr>
<td>s2</td>
<td>The point value.</td>
<td>71.33489150492</td>
</tr>
<tr>
<td>VQT</td>
<td>The point value, quality, and timestamp.</td>
<td>71.334 {Good,13:09:15.32}</td>
</tr>
</tbody>
</table>

The point value (12) is the default, and used for most bindings, while the other options are available if necessary. Certain controls that need to reference timestamps, such as the Trend control, require using VQT.

**Simple Binding - Property Picker**

There are two ways to create a simple binding, either with the Property Picker, or by copy and paste. The following example shows how to use the Property Picker. In this example, we use a hexagon as the reference control, and a rectangle as the linked control.

1. On a blank page, add one hexagon and one rectangle, using the **Simple Path** and **Simple Rectangle** controls.

2. Select the hexagon (the reference control), and in the **Basic Properties**, **Fill Color** choose a color other than **White**, such as **SandyBrown**.

3. Select the rectangle (the linked control), and in the **Basic Properties** click the binding button on the right side of the **Fill Color** row.
A **Binding** selection box will appear, above the color selection dialog.

4. In the **Binding** box, select **Simple**. A text-entry box will appear, and to the right, the Property Picker icon.

5. Click the Property Picker icon. The **Properties** area will turn gray, and when you move the mouse over the page, it will take the shape of the Property Picker.

6. Click the hexagon. This will open the Property Picker menu. From here you can choose which property of the hexagon you wish to give to the rectangle.

7. From the **Basic Properties** submenu, choose **Fill Color**.

The rectangle will change to **SandyBrown**. It is now bound to the color of the hexagon, and will change whenever that color gets changed.
There are two options on the Property Picker menu:

**Filter by Matching Type**
- Reduces the list of properties in the reference control to only those that are also present in the linked control. This helps you quickly identify which properties you can actually bind to this particular control.

**Include Common Properties**
- Hides or displays the Common Properties in the list.

**Simple Binding - Copy and Paste**

Another way to create a simple binding is by copying a reference for a binding and then pasting the link. This is especially useful if you know what properties you need to bind, or if you need to link many controls to a single reference control. Below is an example, based on the controls created in the example above. In this example, the hexagon again is the reference control, and the rectangle is the linked control.

1. Select the hexagon (reference control), and in the **Line Properties**, change the **Stroke Thickness** to 5. You should see the border of the hexagon become 5 pixels thick.

2. Right-click in the text box where you entered 5, and from the drop-down menu, select **Copy Reference**.

3. Select the rectangle (linked control), and in the **Basic Properties, Stroke Thickness, Value** entry field, right click, and from the drop-down menu, select **Paste Link**.
The border of the rectangle will change to 5 pixels. It is bound to the width of the hexagon's border, and will change whenever that gets changed.

Managing Files

File Locations

The various WebView files have different locations, depending on their source and purpose. There are two general locations: for installed content, and for user content; and within user content you'll find your organization content.

Installed Content - [WebRoot]

Installed content is read-only, not to be modified by the user. It is installed (and re-installed) whenever the DataHub is installed, and includes the stock WebView demo pages. Installed content is located here (32-bit/64-bit versions of DataHub):

```
C:\Program Files (x86)\Cogent\Cogent DataHub\plugin\WebServer\html\Content\nC:\Program Files\Cogent\Cogent DataHub\plugin\WebServer\html\Content\n```

We use the placeholder [WebRoot] in this document to refer to this path.

This path, up to html is the default DataHub install directory. It is also the document root directory for the DataHub Web Server, and can be reconfigured in the Properties window Web Server option, if desired.

User Content - [UserContentRoot]

User content such as pages, images, scripts, and so on is read/write, and is located in this directory:
We use the placeholder [UserContentRoot] in this document to refer to this path.

You can change this path using the -H command-line option on DataHub, or by setting it through the DataHub Service Manager application.

Your Organization Content - [OrgLocal]

The path for files pertaining to your organization is:

[UserContentRoot]\Organizations\Local\%

We use the placeholder [OrgLocal] in this document to refer to this path. Within it are sub-directories for various types of files. Here are some examples of types of users and the directories for the files they would typically use:

<table>
<thead>
<tr>
<th>Type of Users</th>
<th>Typically use files in these sub-directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operators</td>
<td>[OrgLocal]\Pages</td>
</tr>
<tr>
<td>Page builders</td>
<td>[OrgLocal]\Pages [OrgLocal]\Images [OrgLocal]\Media</td>
</tr>
<tr>
<td>Advanced/Admin</td>
<td>[OrgLocal]\Branding [OrgLocal]\Scripts</td>
</tr>
<tr>
<td>Control Developers</td>
<td>[OrgLocal]\Controls [OrgLocal]\ControlAssemblies [OrgLocal]\Symbols</td>
</tr>
</tbody>
</table>

There are two possible instances of each of these directories: for shared content, and for user-specific content, and each can optionally have sub-directories.

Shared Content

This is content is available to all logged-in WebView users in your organization. For example:

[OrgLocal]\Pages\TestPage1.xml
[OrgLocal]\Pages\<optional_sub-directory>\TestPage1.xml

User-Specific Content

User-specific content is available only to users logged into WebView with a given user name. This content is placed in the Users sub-directory, and organized in a parallel way to shared content. For example:

[OrgLocal]\Users\<username>\Pages\MyPage1.xml
[OrgLocal]\Users\<username>\Pages\<optional_sub-directory>\MyPage1.xml
And finally, there is one more aspect of user content organization that is relevant only to developers of custom controls.

Custom Controls

Starting with DataHub Version 9, custom controls and control assemblies are differentiated between Silverlight and WPF (Desktop) versions. WebView controls created in Silverlight will only work in Silverlight, and those created in WPF will only work in Desktop WebView. You need to compile these controls separately for Desktop or Silverlight WebView. For example:

```plaintext
[OrgLocal]\Controls\<publisher>\n[OrgLocal]\Controls\<publisher>\Silverlight\n[OrgLocal]\Controls\<publisher>\WPF\n
[OrgLocal]\ControlAssemblies\Silverlight\<publisher>\n[OrgLocal]\ControlAssemblies\WPF\<publisher>\n```

You must similarly distinguish XAML and XML files for your custom controls if they are different between Silverlight and WPF versions.

Editing WebView XML Files

It is possible to edit the XML code in WebView pages. By default, the pages are zipped, so you will need to ensure they are saved as plain text. To do that:

1. Open WebView in Design mode.
2. From the Tools menu, select Options.
3. Scroll to the section Design Mode: Performance (Advanced).
4. Check the setting for Zip files on server and if it is True change it to False.
5. If you changed that setting, then for each page that you need to edit, open the page and then save it again.

You should now be able to make edits to those pages with any text or XML editor.
DataHub WebView Scripting

There are two resources for learning about WebView Scripting:

- For online documentation, please refer to the DataHub WebView Scripting Manual.
- For examples of WebView scripting implementations, please view the last group of these videos, under the title DataHub WebView Video Blog (Advanced Topics), on the Cogent DataHub website.
Dynamic Binding

Using WebView Scripting, it is possible to dynamically bind DataHub points to symbols and controls, and use these capabilities to make template pages.

Dynamic Point Binding

Dynamic point binding allows you to change the DataHub points associated with a control at run time. This tutorial shows two different ways to do this (simple array and dynamic list), using two controls - ComboBox and ListBox.

Combo Box control

**Binding a Combo Box to a Circular Gauge using a simple array**

1. Start the DataPid program from the Windows **Start** menu, the command line, or by clicking on the desktop icon.

   As soon as DataPid starts, it attempts to connect to a DataHub and begins generating data.

2. Open WebView, open a new page, and add a ComboBox control to the page.

3. In the **Basic Properties** of the Combo Box, for the **Items Source**, select the **Script** binding type and edit the default script to read:

   ```
   ['Sp', 'Mv', 'Pv']
   ```

   In this line of script, the `[` and `]` characters tells the WebView scripting engine that this is an array of comma-separated strings, each of which should be assigned to one value in the Combo Box control.
As an alternative to using an array, it is possible to use a script to list the items, as explained in the List Box control section, below.

4. Add a Circular Gauge 2 control to the page.

5. In the Basic Properties of the Circular Gauge 2, for the Current Value, select the Point binding type and make a script:

   ```
   =$DataPid:PID1." + GETP("ComboBox1@SelectedValue");
   ```

When a line of script is entered as a point binding as we see here, the = sign tells the WebView scripting engine that the point name will be assigned. In this example, this allows the name of the point to be constructed by concatenating strings. The GETP function gets the property of a control. The syntax for the GETP argument is a string consisting of the name of the control, an @ symbol, and the name of the parameter that you need to get.

6. Switch to Run mode and choose selections from the Combo Box and see the results in the Circular Gauge.

List Box control

**Binding a List Box to a Trend Chart using a script**

1. Ensure that the DataPid program is running, or start it from the Windows Start menu, the command line, or by clicking on the desktop icon.

2. Open WebView, open a new page, and add a List Box control to the page.
3. In the **Basic Properties** of the List Box, for the **Items Source**, select the **Script** binding type and edit the default script to read:

```javascript
var pts = new List<string>();
pts.Add("Mv");
pts.Add("Pv");
/* More strings can be added ... */
pts;
```

This example script creates a list of strings, and then adds two strings (point names in this case) to the list. The list can hold any number of strings. The final line of the script calls the `pts;` variable, which causes the list of strings to be passed to the List Box as its source of items.

As an alternative to using a script, it is possible to use a an array to list the items, as explained in the **Combo Box control** section, above.

4. Add a Trend Chart (3 pens) control to the page.

5. In the **Basic Properties** of the Trend Chart control, for the **Pen 1 Value**, select the **Point** binding type enter the value `DataPid:PID1.Sp`.

6. For the **Pen 2 Value**, select the **Point** binding type and make a script:

```javascript
="DataPid:PID1." + GETP("ListBox1@SelectedValue");
```
7. Switch to Run mode and choose selections from the List Box and see the results in the Trend Chart.

Dynamic Control and Symbol Binding

Dynamic control and symbol binding allow you to change bindings on a control or symbol at run time. This tutorial shows how to use a Combo Box to change the color of a Shining Light, and then how to change a light switch symbol to appear as if it is being switched on and off.

Control Binding

Bind a Combo Box to a Shining Light

1. Open WebView, open a new page, and add a Combo Box control to the page.
2. In the Basic Properties of the Combo Box, for the Items Source, select the Script binding type and edit the default script to read:

   ```javascript
   WV.GetColorSet();
   ```
3. For **Display Member Path**, choose **Name**.

4. For **Selected Value Path**, choose **Color**.

5. Add a Shining Light control to the page.

6. In the **Basic Properties** of the Shining Light, for the **Primary Light Color**, select the **Simple** binding type and enter the value:

   ```
   ComboBox1@SelectedValue
   ```

7. Switch to Run mode and choose selections from the List Box and see the results in the Trend Chart.
Symbol Binding

Change a Light Switch Symbol Binding to Simulate On and Off

1. Using the same page as above, add a Symbol control.

2. In the Symbol Selection for the Symbol Set, select Symbol Factory, for the Category, select Operator Interface, and for the Symbol, select Light switch (on).

3. Select the ShiningLight control, and in the Basic Properties, right mouse click on Is Light On to copy the reference.

4. Select the Symbol (light switch) and in Custom Events, for OnChecked Event, enter:

```
SETP("ShiningLight1@IsLightOn", true);
```

For this and the next step, the string ShiningLight1@IsLightOn is what
you copied from the Shining Light control. You can simply paste it in.

5. In **Custom Events**, for **OnUnChecked Event**, enter:

   ```
   SETP("ShiningLight1@IsLightOn", false);
   ```

6. Switch to Run mode and click the light switch symbol to turn the Shining Light on and off.

   ![Light Switch](image)

   This works OK, but it looks strange to have the light go off when the light switch is still in the ON position. We'll fix that next.

7. Select the Symbol (light switch) and in **Custom Events**, for **OnChecked Event**, add one more line:

   ```
   SETP("ShiningLight1@IsLightOn", true);
   SETP("@SymbolID", 955272870);
   ```

   For this and the next step, the string `@SymbolID` refers to the symbol itself.

8. In **Custom Events**, for **OnUnChecked Event**, add one more line:

   ```
   SETP("ShiningLight1@IsLightOn", false);
   SETP("@SymbolID", 1685125442);
   ```

9. Switch to Run mode and click the light switch symbol to turn the Shining Light on and off.

   ![Light Switch](image)

   Now the symbol changes from Light switch (on) to Light switch (off) as the light goes on and off.
Creating a Template Page

This tutorial shows how to create a page where you can switch between data sources in a single display, at the click of a button.

Creating an Identical Data Source

1. With DataPid running, start another instance of DataPid. In the second DataPid instance, click the More... button to expose the DataPid Configurable Options.
2. Change the Domain to OtherPid and click the Apply Changes button. Then press the Reconnect button.

If you look in the DataHub Data Browser, you should see a new data domain, OtherPid with data changing values.

At this point, if you are using Desktop WebView you are ready to create the template page, as explained in the procedure below. If you are using Silverlight WebView, you will need to add that data domain for WebView in the DataHub, as explained in the next two steps.

3. Go to the WebView option of the DataHub Properties window, and in the Data Domains Visible to WebView section, click the Refresh button.
4. The domain name OtherPid should appear in the list. Check the checkbox to make the OtherPid domain visible to WebView, and make its points accessible. Now we have two identical point sets with different data to demonstrate our template page.

Creating the Template Page

1. Start WebView, open new page, and add a Combo Box control to the page.
2. In the Basic Properties of the Combo Box, for the Items Source, select the Script binding type and edit the default script to read:

   ```javascript
   [
   "DataPid",
   "OtherPid"
   ];
   ```
3. Add a Point Data Table control to the page.

4. In the **Basic Properties** of the Point Data Table, for the **Point Pattern**, select the **Script** binding type and edit the default script to read:

   \[
   ^\text{"} + \text{GETP("ComboBox1\@SelectedValue")} + ":\text{PID1.} + "\text{..$}";
   \]

   The ^ symbol allows any combination of prefix characters, while the GETP expression pulls in the value from the ComboBox. The ..$ string allows for any combination of suffix characters after the required PID1. string.

5. To make the table easier to read, in the **Table Columns** property, you can make the following columns in the table visible: **Point Name**, **Display Name**, **Value**, and **Quality**.

6. Switch to Run mode and change the data domain in the Combo Box from **DataPid** to **OtherPid** to view the two different sets of data points.

**Adding a Trend Chart to the Page**

1. Add a Trend Chart (3 pens) control to the page.

2. In the **Basic Properties** of the Trend Chart, for the **Pen 1 Value**, select the **Point** binding type and enter:

   \[
   =\text{GETP("ComboBox1\@SelectedValue")} + ":\text{PID1.Sp}";
   \]

   When a line of script is entered as a point binding as we see here, the = sign tells the WebView scripting engine that the point name will be assigned. In this example, this allows the name of the point to be constructed by concatenating strings. The GETP function gets the property of a control. The syntax for the GETP argument is a string consisting of the name of the control, an @ symbol, and the name of the parameter that you need to get.

3. To give a better shape to the trend line for the Sp point, in the **Pen 1 Properties** check the boxes for the **Pen 1 Is Square** and the **Pen 1 Auto Extend** options.
4. Back in the **Basic Properties**, for the **Pen 2 Value** select the **Point** binding type and enter:

\[ \text{=GETP("ComboBox1@SelectedValue") + ":PID1.Mv";} \]

And for the **Pen 3 Value** select the **Point** binding type and enter:

\[ \text{=GETP("ComboBox1@SelectedValue") + ":PID1.Pv";} \]

You should now see all three trends.

5. Switch to Run mode, and again change the data domain in the Combo Box from **DataPid** to **OtherPid**. Now both the table and the trend chart alternate between the two different sets of data points.

Following this example, you can build a page to display multiple identical sets of data in any group of controls, with the ability to switch between data sets at the click of a button.
Customizing DataHub WebView

Long strings in some examples have been wrapped. The code examples in this chapter, and possibly others, may include very long strings that do not render well in the printed output of this documentation. For this reason, we have wrapped those strings in the examples. Please be aware that those examples may not work properly in your code if you copy them verbatim. You may need to unwrap the strings.

Simple Branding

Simple branding allows a WebView administrator to create a branded login page, to specify the application title, and to add company-specific links to the Help menu. Simple branding is intended for site administrators who want to provide their users with a general sense that the application is an integral part of the company's software solution for systems and processes.

In addition to simple branding options, OEM branding allows a distributor to replace Cogent-specific logos and references with icons and text of its own choosing. This level of branding is intended for Cogent partners and licensed distributors who need to re-brand the application to leverage their own corporate brand and to capitalize on specific market opportunities. For more information on OEM Branding, please contact Cogent.

Prerequisites

• An understanding of XML.
• Familiarity with XAML.
• Access to [UserContentRoot].

Creating a custom login page

1. Using XAMLPad (or your favorite XAML editor), create a standalone XAML file that contains the details of your login page. The file must be named LoginPage.xaml and stored in a subdirectory of the [OrgLocal]\Branding directory. By default, WebView will look in [OrgLocal]\Branding for your branding information. Alternatively, you can use the DataHub WebView Configuration properties dialog to set a specific custom branding directory.

Your page may contain any XAML you wish, including graphics, controls and animation. For your convenience, you may wish to start by copying the sample LoginPage.xaml file from the [WebRoot]\Common\Branding\Template subdirectory.

Images must be either JPEG (*.jpg) or PNG (*.png) format.

2. Your page must include a DataHubConnection element (and associated XML namespace). This element provides support to authenticate the user (i.e., username and
password). Even if you want to bypass the login page with automatic credentials, this element must be included in LoginPage.xaml.

To add the DataHubConnection element:

a. At the top of LoginPage.xaml, on the root element, add an XML namespace reference to the DataHubWebViewApplicationInfrastructure assembly:

```xml
<Grid
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:dhc="clr-namespace:Cogent.DataHubWebView;assembly=DataHubWebViewApplicationInfrastructure"
```

(Long strings have been wrapped [56])

b. Insert the DataHubConnection element where you want to place the credential prompt, for example:

```xml
<dhc:DataHubConnection Background="Transparent"/>
```

When the user navigates to your published ASP/HTML URL to launch WebView, the DataHubConnection element will appear:

![DataHubConnection element](image)

Simple branding does not support customizing the style of this element. You can set visibility, margin and other standard attributes, but you cannot change the text, colors or button style.

### Specifying text, icon, and URL targets

1. Using your favorite editor, create a standalone XML file to contain the branding settings for the application. The file must be named Branding.xml and stored in a sub-directory of the \OrgLocal\Branding directory.

2. For your convenience, you may wish to start by copying the sample Branding.xml file from the \WebRoot\Common\Branding\Template\ directory.

```xml
<c:Branding
    xmlns:c="clr-namespace:Cogent.DataHubWebView;assembly=DataHubWebViewApplicationInfrastructure"

    ApplicationName="Your Application Title"
    CompanyName="Your Company Name"
```
Customizing DataHub WebView

```xml
<WebBrowser>
  <CompanyImageSource>Branding/Template/CompanyIcon.png</CompanyImageSource>
  <CompanyUrl>http://www.yourcompany.com</CompanyUrl>
  <OnlineHelpUrl>http://www.yourcompany.com/Docs/ProductOverview.html</OnlineHelpUrl>
</WebBrowser>
```

(Long strings have been wrapped [56])

3. Do not change the XML namespace references (`xmlns` and `xmlns:c`) or the names of the Branding attributes (e.g., `ApplicationName`).

4. Replace the placeholder text and URLs with your branding information:
   - `ApplicationName`: incorporated into the browser title
   - `CompanyName`: incorporated into the Help menu's About item text.
   - `CompanyImageSource`: the icon for the Help menu's About item text.
   - `CompanyUrl`: the target for the Help menu's About item text.
   - `OnlineHelpUrl`: the target for the Help menu's About item text.

5. Do not change the `Version` attribute or its value.

Adding a favorite icon

   
   If found, this icon is used in several places to represent your website, such as the browser address bar, Favorites list, etc.
2. To use your own icon, replace the WebView default `favicon.ico` file deployed during a typical installation.

For more information on `favicon.ico`, see http://en.wikipedia.org/wiki/Favicon.

**Testing the results**

1. Ensure your `Branding.xml` and `LoginPage.xaml` files (and all local resources referenced by the login page) have been copied to the `[OrgLocal]\Branding` directory.

2. Launch WebView. It should present your custom login page. After entering your credentials, you should see that the application references the settings you specified in the `Branding.xml` file.

   In the Cogent DataHub Properties window, WebView option, there is a checkbox called **Use a custom branding directory**. In previous DataHub versions, this box needed to be checked. This is no longer necessary.

**Initialization Parameters**

DataHub WebView offers a number of initialization parameters that give you some control over the connection, log-in, initial view of a page, and designer controls when a page gets loaded. How to use these parameters depends on whether you are using Desktop WebView or Silverlight WebView.

**Desktop WebView**

For Desktop WebView you can put any of the initialization parameters into the properties of desktop shortcut that you use to launch WebView. Each of the two scenarios for running Desktop WebView has its own desktop shortcut, which must be configured separately.

1. **Running Locally** Create a desktop shortcut for Desktop WebView if you don't already have one, and right-click on it to expose the **Target** properties.
2. **Web Launch** Create a desktop shortcut for Desktop WebView if you don’t already have one as explained here, and right-click on it to expose the **URL** properties.

For either of these, you can add parameters by inserting a space, followed by the parameter, an equal sign (=), and the value. Multiple parameters are allowed. For example:

```
StartInRunMode=true Page="Circular Gauges"
```

```
StartInRunMode=true Page=Notifications
```

Shortcuts for a locally running DataHub can contain spaces, as long as the value is in quotation marks, such as "Circular Gauges" above. Web launch shortcuts that use URLs do not allow spaces in the value. It must be a single string, as in the second example above (Notifications).

Please refer to the Parameter List to see all of the available parameters.

**Silverlight WebView**

Silverlight WebView is implemented as a Microsoft Silverlight object embedded in a web page. Consequently, it is possible to apply formatting and parameters to the `<object>` tag.
in the HTML of that web page. This allows the administrator of the WebView installation to customize the user experience, for example to set background color, size and position of the WebView object within the page. In addition it is possible to pass parameters to WebView from the web page to customize the behaviour of the WebView application. This document deals with these various initialization parameters.

The **initParams** Parameter

The Silverlight object recognizes a parameter on the `<object>` tag called *initParams*. In its simplest form, it is specified in HTML like this:

```html
<object id="DataHubWebView" ...>
  <param name="initParams" value=""/>
</object>
```

The values of the initialization parameters can be inserted into the `value` attribute of the `<param>` tag as a comma-separated list of parameters. Each parameter consists of a parameter name, followed by the equal sign, (=), followed by the parameter value.

**Accessing Parameters from the DataHub Properties Window**

The DataHub Properties window allows you to specify some of the possible initialization parameters that are accepted by WebView. You can access these parameters in HTML as follows:

1. Ensure that the HTML page containing the WebView object has the extension `.asp` instead of `.html`. This will allow the Cogent DataHub to execute Gamma scripts specified in the page when the web client loads the page. The default URL for starting WebView is `http://localhost/Silverlight/DataHubWebView.asp`.

2. Make a copy of the page `DataHubWebView.asp`, and make all edits to that copy. You will need to point to that copy in your URL.

   **⚠️** If you edit the original page you will lose your edits whenever WebView is updated.

3. Add the following HTML to the new page:

   ```html
   <%
   require ("WebViewSupport.g");
   local initparams = WebViewInitString();
   %>
   ```

4. Modify the *initParams* in the WebView `<object>` tag as follows:

   ```html
   <object id="DataHubWebView" ...>
     <param name="initParams" value="<%= initparams %>"/>
   </object>
   ```

   Every time the page is loaded, the WebView *initParams* will be computed from the settings in the DataHub Properties window.
Adding Custom Parameters

To customize the initialization parameters, simply add your own definitions to the `init-Params` `<param>` tag. For example:

```xml
<param name="initParams" value="Page=MyStartPage,StartInRunMode=true" />
```

You may combine your own custom parameters with those specified in the DataHub Properties window simply by adding your own parameters to those read from the Properties window. For example:

```xml
<param name="initParams" value="<%= initparams %>,Page=MyStartPage,StartInRunMode=true" />
```

The last value of a parameter will take precedence, so if you insert your parameters after the `<%= initparams %>` ASP code, as in the example above, then your parameters will override those specified in the Properties window.

Specifying Parameters in the Page URL

Some WebView initialization parameters can be specified in the URL of the web page, starting with a question mark (`?`). Multiple parameters are separated by ampersands (`&`). For example:

```
http://localhost/Silverlight/DataHubWebView.asp
?StartInRunMode=true
```

```
http://localhost/Silverlight/DataHubWebView.asp
?username=admin&password=admin
```

The URL columns in the tables below indicate which parameters can be specified in this way.

Parameter List

The following tables show all of the initialization parameters available in WebView. The parameter names are not case sensitive, but the parameter arguments may be, depending on the meaning of the argument. The Properties column indicates whether the parameter can be set via a selection in the Cogent DataHub Properties window. The URL column indicates whether the Silverlight WebView parameter can be set in the URL for the web page.

Table 1. Connection

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Properties</th>
<th>Available</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcpPort</td>
<td>Integer</td>
<td>Advanced</td>
<td>Desktop</td>
<td>N/A</td>
<td>The port number used to connect to the DataHub data feed. The default is 4502.</td>
</tr>
</tbody>
</table>
### Customizing DataHub WebView

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Properties</th>
<th>Available</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebPort</td>
<td>Integer</td>
<td>Advanced</td>
<td>Desktop</td>
<td>N/A</td>
<td>The HTTP port number on which the DataHub Web Server is listening.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options</td>
<td></td>
<td></td>
<td>The default is 80.</td>
</tr>
<tr>
<td>Host</td>
<td>String</td>
<td>Advanced</td>
<td>Desktop</td>
<td>N/A</td>
<td>The name or IP address of the computer on which the DataHub is running.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataProtocol</td>
<td>String</td>
<td>Advanced</td>
<td>Desktop</td>
<td>N/A</td>
<td>There are two options: &quot;TCP&quot; or &quot;WebSocket&quot; to specify the data protocol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options</td>
<td></td>
<td></td>
<td>The default is &quot;TCP&quot;.</td>
</tr>
<tr>
<td>UseSsl</td>
<td>True/False</td>
<td>Advanced Options</td>
<td>Desktop</td>
<td>N/A</td>
<td>When true, WebView will use the SSL protocol for the TcpPort. The default is false.</td>
</tr>
<tr>
<td>UseHttps</td>
<td>True/False</td>
<td>Advanced Options</td>
<td>Desktop</td>
<td>N/A</td>
<td>When true, WebView will use HTTPS for the WebPort. The default is false.</td>
</tr>
</tbody>
</table>

**Table 2. Credentials and Login**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Properties</th>
<th>Available</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>True/False</td>
<td>Running WebView</td>
<td>Desktop, Silverlight</td>
<td>No</td>
<td>This entry is hard-coded as Local, but can be overridden for connections to SkkyHub.</td>
</tr>
<tr>
<td>UserName</td>
<td>String</td>
<td>Running WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>The user name. If this is specified then Silverlight WebView will bypass the login screen and automatically log in using this user name.</td>
</tr>
<tr>
<td>Password</td>
<td>String</td>
<td>Running WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>The password for UserName.</td>
</tr>
</tbody>
</table>

**Table 3. Start Page, Page Data, and Initial Editor Mode**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Properties</th>
<th>Available</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>String</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>The WebView page to show when the user first logs in. This page will be loaded even if the user starts in Design mode. The page name must include any path components separated by / characters. The page name does not include an extension.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Properties</td>
<td>Available</td>
<td>URL</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------</td>
<td>------------</td>
<td>-----------</td>
<td>-----</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PageData</td>
<td>name=value pairs</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>These values become global variables in the script context. Each name=value pair is separated by a comma, as in: PageData=a=5,b=6. Please see Page Data in the WebView Scripting manual for more information.</td>
</tr>
<tr>
<td>StartInRunMode</td>
<td>True/False</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, WebView will enter Run mode when the user logs in, otherwise it will enter Design mode if the user has permission to do so.</td>
</tr>
<tr>
<td>DisableDesignMode</td>
<td>True/False</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, WebView will not allow the user to enter Design mode, even if the user has permission to do so. Also, when true, this parameter hides dialogs that require user interaction.</td>
</tr>
<tr>
<td>UseKioskView</td>
<td>True/False</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, WebView will start in Kiosk mode, removing the menu and icon bars.</td>
</tr>
<tr>
<td>UseFullScreenMode</td>
<td>True/False</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, WebView will start in Full Screen mode, displaying only the page.</td>
</tr>
<tr>
<td>HideInitializationElements</td>
<td>True/False</td>
<td>No</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, no wallpaper is displayed during initialization of the WebView application.</td>
</tr>
<tr>
<td>RunSilently</td>
<td>True/False</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, in Run mode only, this parameter hides dialogs that require user interaction. The default is false.</td>
</tr>
<tr>
<td>HideLoadingPageMessage</td>
<td>True/False</td>
<td>DataHub WebView</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, in Run mode only, this parameter hides dialogs that do not require user interaction (such as any &quot;Wait&quot; dialog). The default is false.</td>
</tr>
</tbody>
</table>
### Customizing DataHub WebView

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Properties</th>
<th>Available</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceNonInteractiveDialogs</td>
<td>True/False</td>
<td>No</td>
<td>Desktop,</td>
<td>Yes</td>
<td>When true, in Run mode only, this parameter overrides the RunSilently parameter and displays non-interactive dialogs (such as any &quot;Wait&quot; dialog). The default is false.</td>
</tr>
<tr>
<td>FreezeScreenWhileLoadingPage</td>
<td>True/False</td>
<td>No</td>
<td>Silverlight</td>
<td>Yes</td>
<td>When true (the default), then the current page is frozen and the new page is loaded in the background. If this is false then the current page is erased and the new page is loaded in the foreground, with controls on the screen appearing as they load.</td>
</tr>
<tr>
<td>ShowExitConfirmation</td>
<td>UnsavedChanges, Never, Always</td>
<td>No</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>This setting determines whether to prompt the user when he attempts to exit WebView. If set to Never, then the user will not be prompted on exit. If set to Always then the user will be prompted. If set to UnsavedChanges then the user will only be prompted if there are unsaved changed on the current page. The default is Always.</td>
</tr>
</tbody>
</table>

### Table 4. Branding

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Properties</th>
<th>Available</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branding-Folder</td>
<td>String</td>
<td>DataHub WebView</td>
<td>Desktop,</td>
<td>Yes</td>
<td>The path to a folder to search for custom branding information. This folder is relative to the WebView installation's Branding directory. The path separator is the / character.</td>
</tr>
</tbody>
</table>
Table 5. Page Designer Functionality

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Properties</th>
<th>Available</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisablePageInformationButton</td>
<td>True/False</td>
<td>DataHub</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, the page information button will not be displayed in Run mode. The default is false.</td>
</tr>
<tr>
<td>MakeDataPointsReadOnly</td>
<td>True/False</td>
<td>DataHub</td>
<td>Desktop, Silverlight</td>
<td>Yes</td>
<td>When true, WebView will be unable to write data to the DataHub's data set, regardless of the configuration or user permissions. The default is false.</td>
</tr>
<tr>
<td>ShowHiddenControls</td>
<td>True/False</td>
<td>No</td>
<td>Desktop, Silverlight</td>
<td>No</td>
<td>When true, controls that are normally hidden will be visible in Design mode. Hidden controls are controls that act as base classes for other controls, are deprecated, or are experimental. In any case, these controls should not be used by a page designer. The default is false.</td>
</tr>
</tbody>
</table>

Adding Controls

This section outlines the steps to add a custom control, with custom behavior, that will be used inside the WebView application.

This procedure is intended for control designers and power users who want to complement or extend the suite of controls that ship with WebView. For those who merely want to re-style an existing control, please refer to a soon-to-be-added section: Restyling a WebView Control.

Prerequisites

- An understanding of how to develop Microsoft .NET class libraries.
- Familiarity with Visual Studio 2010.
- An understanding of XML.
- Some familiarity with XAML.
- Access to the Cogent DataHub installation directories and files.

Preparing the Visual Studio project

The first step in adding a custom control is to create a Visual Studio project containing the default XAML style, ControlTemplate, and the control's code-behind class.
1. In Visual Studio 2010, create a new project:
   - Template: **Silverlight Class Library**
   - Version: **Silverlight 4**
   - Project Name: **MyGauges**
   replacing **MyGauges** with the name of your project.

2. Delete `class1.cs` and add a new class to represent your control. For this documentation, we name the class **GlossyGauge**; which you can replace with your own class name.

   ```csharp
   public class GlossyGauge
   ```

3. Now the XAML needs to be separated from the code-behind class. Create a new directory and add an XAML file to that directory, as follows:
   - New directory name: **Themes**
   - New XML file name: **Generic.xaml**

4. Replace the content of `Generic.xaml` with the following XAML code:

   ```xaml
   <ResourceDictionary
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       xmlns:vsm="clr-namespace:System.Windows;assembly=System.Windows"
       xmlns:local="clr-namespace:MyWebViewControls.MyGauges"

       >

       <Style TargetType="local:GlossyGauge">
       <Setter Property="Template">
       <Setter.Value>

       <ControlTemplate TargetType="local:GlossyGauge">
       <Grid x:Name="ControlRoot"
           Background="Transparent">
       <Viewbox>

   ```
5. Set the local XML namespace to match the namespace used for your control (i.e., where the *GlossyGauge* class resides).
   ```xml
   xmlns:local="clr-namespace:MyWebViewControls.GlossyGauge"
   ```

6. Set the TargetType (for both the *Style* and *ControlTemplate* elements) to refer to your class:
   ```xml
   TargetType="local:GlossyGauge"
   ```

7. Between the start and end `<Viewbox>` tags, paste the XAML code that represents your control (perhaps created using Microsoft Expression Blend).

8. Change the definition of your class to work with the *Style* and *ControlTemplate* (in *Generic.xaml*) by inheriting from *Control* and by using the partial modifier.
   ```csharp
   public partial class GlossyGauge : Control
   ```

9. Add a default constructor to your class and set the *DefaultStyleKey* property to refer to the class's type. This triggers the framework to invoke the *OnApplyTemplate* method and apply the matching XAML template in the project's `Themes\Generic.xaml` file.
   ```csharp
   public GlossyGauge()
   {
       this.DefaultStyleKey = typeof(GlossyGauge);
   }
   ```

10. For illustration, add a Dependency Property, which will manage the *Background-Color* of the control, and will be exposed to the user via Property Explorer.
    
    The *UpdateVisualState* method will be coded in an upcoming step.
    ```csharp
    public Color BackgroundColor
    {
        get { return (Color)GetValue(BackgroundColorProperty); }
        set { SetValue(BackgroundColorProperty, value); }
    }
    ```
    ```csharp
    public static readonly DependencyProperty BackgroundColorProperty = DependencyProperty.Register("BackgroundColor", typeof(Color), typeof(DHCommonGauge), new PropertyMetadata(Colors.Black,
```
new PropertyChangedCallback(DependencyPropertyChanged)));

private static void DependencyPropertyChanged(
    DependencyObject sender, DependencyPropertyChangedEventArgs e)
{
    DHCommonGauge gauge = sender as DHCommonGauge;
    gauge.UpdateVisualState();
}

11. Define the ControlTemplate behavior, i.e., the interaction among the Style and ControlTemplate you defined in Generic.xaml and your control's properties and behavior.

private Grid ControlRoot;
public override void OnApplyTemplate()
{
    // Capture the Framework Elements from the ControlTemplate
    ControlRoot = GetTemplateChild("ControlRoot") as Grid;

    UpdateVisualState();
}

private void UpdateVisualState()
{
    // Logic to update the control, e.g., gauge range,
    // current value, color, etc.
    // For example:
    ControlRoot.Background =
        new SolidColorBrush(this.GaugeBackgroundColor);
}

For example, the above code:

- Defines the ControlRoot element, which is set in the OnApplyTemplate method by referencing the named element in the ControlTemplate.
- Updates the Background of the ControlRoot whenever the UpdateVisualState method is called, including when the BackgroundColor Dependency Property changes.

For more information about modifying custom control behavior, please refer to a soon-to-be-added section: Making your Custom WebView Control Host-Aware.

The XAML file to reference the control

You will need to create a standalone XAML file that references the control.

1. Using your favorite editor, create a standalone XAML file that references the control you created in your Visual Studio project. The file can be named anything you like. By convention, the file name refers to the class (e.g., GlossyGauge.xaml).

    The root element is <UserControl> and four XML namespaces are required:
• xmlns and xmlns:x are used to process XAML.
• xmlns:live is used by the WebView infrastructure to associate XAML elements with control Parameters (discussed in the next section).
• xmlns:myControls is used to reference your Visual Studio control.

```xml
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:live="clr-namespace:Cogent.DataHubWebView;assembly=DataHubWebViewApplicationInfrastructure"
    xmlns:myControls="clr-namespace:MyWebViewControls.MyGauges;assembly=MyWebViewControls"
>
    <myControls:GlossyGauge live:DynamicElement.ID="Gauge" />
</UserControl>
```

(Long strings have been wrapped [56])

2. When ready to test, this file needs to be put into the appropriate Controls directory, as explained in the Custom Controls section of File Locations.

**The XML file for public properties and behavior of the control**

In this section, we will create a standalone XML file that encapsulates the public properties and behavior of the control.

1. Using your favorite editor, you need to create a standalone XML file to define the public interface for your WebView control. The file can be named anything you like. By convention, the file name refers to the class (e.g., GlossyGauge.xml). The file must end with the .xml file extension.

   The best way to begin is to use one of the existing control XML files as a template. These files are located in the [WebRoot] directory. You should copy it into the appropriate Controls directory, as explained in the Custom Controls section of File Locations, and make all of your edits to that copy.

2. WebView parses control XML files to determine:

   • Identifying control information, including ControlID (unique), ControlName, Description and ShortDescription (the latter three are presented to the user in Property Explorer).
   • The base XAML used to render the control, ControlFileName, which in turn references your control class.
   • The public data (i.e., Parameters) to make available to the user via Property Explorer. Parameters are grouped within ParameterCategory elements.

   For example, the XML file for the GlossyGauge control might look like this:

   ```xml
   <c:EditableObjectConfig Owner="System" Category="Gauges"
   ```
Customizing DataHub WebView

ControlID="MyControls:GlossyGauge:1" ControlName="My Glossy Gauge"
ControlFileName="GlossyGauge.xaml"
Description="This gauge has glossy style and looks cool!"
ShortDescription="Cool-looking glossy gauge"
ToolbarImageFileName="GlossyGauge.png"
Width="200" Height="200">
</c:EditableObjectConfig.ParameterCategories>
</c:EditableObjectConfig.ParameterCategories>
</c:EditableObjectConfig>

3. Notes for Parameters:

- The c: namespace prefix is used by WebView and is required for the EditableObjectConfig, ParameterCategory and Parameter elements.

- ParameterName is required and must be unique for the control.

- Label values can be any text you want. They appear in Property Explorer.

- WebView uses EditorType to determine how the user should interact with the Parameter value in Property Explorer. EditorType can be one of several values, including bool, color, enum, togglebutton, etc.

- The Default value is used as the initial Parameter value.

- TargetPath refers to the source object and source property path of the control to which this Parameter maps. For example, referring to:

```xml
<Parameter ParameterName="BackgroundColor" ...
  Default="#80FF0000" TargetPath="Gauge.CurrentValue" />
```
the **BackgroundColor** Parameter has a **TargetPath of Gauge.BackgroundColor** which means it maps to:

- the XAML element defined with the **DynamicElement.ID** of **Gauge** i.e., the element defined in **GlossyGauge.xaml** (ControlFileName)
  ```xml
  <myControls:GlossyGauge live:DynamicElement.ID="Gauge" />
  ```
- and the **"BackgroundColor" Dependency Property** defined in the control class.

- There are several more attributes that can be used to configure the **EditableObjectConfig** and **Parameter** objects. For a comprehensive list, please refer to "Configuring Control XML with EditableObjectConfig and Parameter Attributes."

**Testing the results**

1. Confirm your Visual Studio project (which includes the **GlossyGauge** class and **Generic.xaml**) builds successfully.
2. Copy the resulting assembly (**MyGauges.dll**) into the appropriate **ControlAssemblies** directory, as explained in the **Custom Controls** section of File Locations.
3. Confirm that **GlossyGauge.xaml**, **GlossyGauge.xml** and **GlossyGauge.png** reside in the corresponding Controls directory.
4. Launch WebView. Your control should be listed in Control Explorer and on the Controls Toolbar. You should be able add an instance of your control to any page.
5. Now that you’ve confirmed you can build a control and get it working inside WebView, it’s time to enhance and extend your control’s functionality. It’s a simple, iterative cycle:
   - Define new **Dependency Properties** on your control class.
   - Implement new behaviors with custom functionality in your control class and with the control XAML in the **GlossyGauge.xaml** file.
   - Build your class library and copy the **dll** to the **ControlAssemblies** installation directory.
   - In the control XML file (**Controls installation directory**), add new **Parameters** for each of the control properties you want to expose to the user.
Creating Custom Symbols

WebView uses a single "Symbol Host" control to host all of the available symbols. Symbols are arranged in symbol sets, where each symbol set has one or more categories, and each category has one or more symbols. The symbols themselves are just drawings in their simplest form. Consequently, you can add different symbols to WebView by creating your own drawings and telling WebView where to find them.

What you'll need

1. A XAML Editor. Symbols are specified entirely in XAML. The XAML editor can be any one of the following:
   - Microsoft Expression Blend
   - Microsoft Visual Studio
   - Any text editor (e.g., Notepad++)

   Microsoft Expression Blend is the most powerful of these tools. Microsoft Visual Studio 2010 Express is free of charge, as are many text editors. We do not recommend writing XAML using a text editor until you are very experienced.

2. An XML Editor. Symbol maps are specified in XML. A symbol map tells WebView how to identify your symbols and how to associate the XAML files with the symbol names and identifiers. XML is a plain-text format that can be edited with any text editor, including the editors built into Expression Blend and Visual Studio.

Creating Your Symbol Library

Create a symbol map

The symbol map is a simple XML file that tells WebView about your symbol set. The first part details information about the whole symbol set, and the second part identifies the individual symbols. Below is an example of a symbol map file. A symbol map file can be formatted either as 7-bit ASCII or as UTF-8. For example, the following map file defines two shapes, an ellipse and a rectangle:

```xml
<?xml version="1.0" encoding="utf-8"?>
<wvsm:SymbolMap
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:wvsm="clr-namespace:Cogent.DataHubWebView;assembly=DataHubWebViewSymbolHelpers"

 SymbolSet="My Test Symbols"
 Manufacturer="Cogent Real-Time Systems Inc."
 Copyright="(C) 2011 Cogent Real-Time Systems Inc."
 License="Licensed for use with DataHub WebView only."
```
Creating Custom Symbols

| Description | These Symbols illustrate how to incorporate custom objects into DataHub WebView. |
| SymbolSetVersion | 1.0 |

```xml
<wvsm:SymbolMap.Symbols>
  <wvsm:Symbol Category="Simple Shapes" Name="Rectangle" SymbolID="201" FileName="SimpleShapes.xaml" Style="RectangleStyle" />
  <wvsm:Symbol Category="Simple Shapes" Name="Ellipse" SymbolID="202" FileName="SimpleShapes.xaml" Style="EllipseStyle" />
</wvsm:SymbolMap.Symbols>
```

(Long strings have been wrapped [56])

You can make a copy of this symbol map and alter it to your needs. Some of the Symbol-Map fields deserve further description:

**SymbolSet**

The name that will appear in the WebView user interface when the user chooses to place one of your symbols on the screen.

**SymbolSetVersion**

A version number of the form *N.N*. The Symbol Host control can interpret this version number to allow you to offer multiple versions of the same symbol. The intention here is that you could create an equivalent visual representation of your symbols and make them available as an optional upgrade from a previous version. Different versions of symbols should perform the same role, such that a newer version of a symbol is a "drop-in" replacement of the previous version.

In addition to the symbol set definitions, this file contains a Symbols list, with a single entry for each symbol in the symbol set. Each symbol has the following fields:

**SymbolID**

A number that must be unique for each symbol in this symbol set. This is the most important identifying element of a symbol. Once you have released your symbol set, you must not change this number for a given symbol. If you release multiple versions of your symbol set, then the SymbolIDs must match for different versions of the same symbol. SymbolID is a 32-bit signed integer number, and does not have to be in order or contiguous.

**Category**

A character string that identifies the symbol category. This category will be displayed to the user in the WebView interface. You may use any string you like. If two or more symbols have the same category name, they will be grouped within the WebView interface.

**Name**

The symbol name that will be displayed to the user in the WebView interface. This
Creating Custom Symbols

name does not have to be unique, as the SymbolID is the sole unique identifier for a symbol.

FileName
This is the name of the file that contains the symbol's XAML definition. When WebView encounters this symbol, the XAML file will be read from the server. WebView will only look for the file in the same directory as it originally found your symbol map file.

Style
This identifies the <Style> tag associated with this particular symbol within the XAML file. You may combine multiple symbol definitions into a single XAML file. The specific XAML definition of a symbol is identified by a <Style> tag within that file.

Create a symbol XAML file

The symbol file defines a single UserControl. The definition of that control consists of a header, one or more ControlTemplate resources, a matching number of Style resources, and an instance definition of a DHSymbol control. In its simplest form, it would look like this:

```xml
<UserControl
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 mc:Ignorable="d"
 xmlns:sym="clr-namespace:Cogent.DataHubWebView.Controls;assembly=DataHubWebViewSymbol"
>
  <UserControl.Resources>
    <ControlTemplate x:Key="RectangleTemplate" TargetType="sym:DHSymbol">
      <Grid>
        <Rectangle Fill="Transparent" Stroke="Black"/>
      </Grid>
    </ControlTemplate>
    <ControlTemplate x:Key="EllipseTemplate" TargetType="sym:DHSymbol">
      <Grid>
        <Ellipse Fill="Transparent" Stroke="Black"/>
      </Grid>
    </ControlTemplate>
    <Style x:Key="RectangleStyle" TargetType="sym:DHSymbol">

```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
Creating Custom Symbols

A WebView symbol can contain any legal XAML definition. That means that a symbol is in fact more than just a drawing. It could contain buttons, dials, shapes, storyboards and virtually anything that XAML can describe. However, you may not provide C# code-behind class for your symbol. The intention is that the symbol definition contains the rendering definition of the symbol, while the behaviour of the symbol is implemented by the Symbol Host control.

The symbol XAML file can contain any number of matching pairs of <ControlTemplate> and <Style> tags. The <Style> tag for a particular symbol refers to its matching <ControlTemplate> in the Template property within the Style.

Deploy your symbol set

By default, the base directory for all installed symbols is [WebRoot]\Common\Symbols\. You deploy your custom symbol set by simply copying the XAML files and symbol map file from that directory into a sub-directory of [OrgLocal]\Symbols. This sub-directory is specific to your symbol set.

For example, you might create a directory called:

[OrgLocal]\Symbols\Test Symbols V1.0

Then you would copy your symbol files into this directory. Each symbol file should have a .xaml filename extension, and the symbol map must have a .map extension, like this:

[OrgLocal]\Symbols\Test Symbols V1.0\SimpleShapes.xaml
[OrgLocal]\Symbols\Test Symbols V1.0\Test Symbols V1.0.map

Changing symbol files or the symbol map requires all users to restart WebView.
Symbol Animation

In its simple form a symbol is not animated. That is, it does not blink, rotate or change color as its input data value changes. In order to animate your symbol you must indicate which parts of your symbol will change according to its data value. The Symbol Host control provides XAML markup attributes that you can use on the parts of your symbol to produce the animation of your choice. For example, if you include an attribute on a part of your symbol that indicates blinking behaviour, the Symbol Host control will enable the blinking properties for that symbol and will allow your user to configure the conditions under which the symbol will blink.

Animation attributes

An animation attribute is defined as an XML attribute attached to any UIElement tag within your symbol's XAML definition. For example, to make our ellipse blinkable, we can modify its template definition:

```xml
<ControlTemplate x:Key="EllipseTemplate" TargetType="sym:DHSymbol">
    <Grid sym:DHSymbol.BlinkTargetProperty="Opacity">
        <Ellipse Fill="Transparent" Stroke="Black" />
    </Grid>
</ControlTemplate>
```

In this case, we have attached the blink behaviour to the top-level UIElement (the Grid) in the symbol. This will cause the entire symbol to blink when it is configured to do so. We have chosen to blink the symbol by modifying the Opacity property of the Grid, which in practice is the only reasonable property to use for the purpose.

We may not want to blink the entire symbol. For example, the following XAML code modifies the ellipse to contain a rectangle within it:

```xml
<ControlTemplate x:Key="EllipseTemplate" TargetType="sym:DHSymbol">
    <Grid>
        <Ellipse Fill="Transparent" Stroke="Black" />
        <Rectangle Fill="Red" Stroke="Transparent" Width="20"
            sym:DHSymbol.BlinkTargetProperty="Opacity"/>
    </Grid>
</ControlTemplate>
```

Now the ellipse outline will remain solid, and only the rectangle within it will blink.

You may add more than one animation attribute to a single element within your XAML definition, and you may add the same animation attribute to more than one element. Thus, if you have a symbol consisting of many elements, you can choose on an individual basis which element would blink, change color etc.

For example, we can add a rotation attribute to our rectangle:
Creating Custom Symbols

Note that the animation attributes do not immediately animate the symbol. They indicate to the Symbol Host control that this symbol will participate in that type of animation according to the configuration that the user has applied. The symbol designer might offer a rotation animation in the symbol, but the page developer may choose not to enable it for his particular configuration.

Your symbol may implement a number of user-definable colors. These can simply be added to any element of the symbol's XAML code. For example, to make the base color of the ellipse into a user-definable property, we can add the BaseColorTargetProperty attribute:

```xml
<ControlTemplate x:Key="EllipseTemplate" TargetType="sym:DHSymbol">
    <Grid>
        <Ellipse Fill="Transparent" Stroke="Black" /
        <Rectangle Fill="Red" Stroke="Transparent" Width="20"
            sym:DHSymbol.BlinkTargetProperty="Opacity"
            sym:DHSymbol.RotationType="Normal"
            />
    </Grid>
</ControlTemplate>
```

In this example, we add tell the Symbol Host control that the base color property should be applied to the Fill.Color property of the Ellipse. The Fill.Color refers to the Color of the SolidColorBrush assigned to the Fill property of the Ellipse. Since this reference requires Fill to be a SolidColorBrush, we must also assign Fill to be initially Transparent (or any solid color brush). Now when the user modifies the Base Color property of the Symbol Host in the Aesthetics property category, the fill color of the Ellipse within our symbol will change.

**Animation types**

The Symbol Host control offers several animation types. A symbol designer may specify any combination of these types within his symbol, or may choose to implement none at all. These types are:
### Type: Constant

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseColorTargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will change color when the Base Color is altered. E.g., a Rectangle's Fill or a Grid’s Background.</td>
</tr>
<tr>
<td>AccentColor1TargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will change color when the Accent Color 1 is altered.</td>
</tr>
<tr>
<td>AccentColor2TargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will change color when the Accent Color 2 is altered.</td>
</tr>
<tr>
<td>ShadowColorTargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will change color when the Shadow Color is altered.</td>
</tr>
<tr>
<td>GlassinessTargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will receive a value in the range 0.0 - 1.0 in response to the Glassiness symbol property. This is normally used to modify the opacity of a gradient overlay on the symbol that will provide the appearance of a glass cover.</td>
</tr>
</tbody>
</table>

### Type: Condition

Condition specifies that part of the symbol will change its color, text or value based on the input value condition.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConditionColorTargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will change color when the Condition Color is altered.</td>
</tr>
<tr>
<td>ConditionColorOpacityTargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will receive a value in the range 0.0 - 1.0 in response to the Condition Color Opacity symbol property.</td>
</tr>
<tr>
<td>ConditionTextTargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of the control property that will receive</td>
</tr>
</tbody>
</table>
Creating Custom Symbols

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlinkTargetProperty</td>
<td>String (a Property Name)</td>
<td></td>
<td>The name of the property that will receive a value between 0.0 - 1.0 indicating the blink opacity.</td>
</tr>
<tr>
<td>BlinkColorTargetProperty</td>
<td>String (a Property Name)</td>
<td></td>
<td>The name of the property that will receive a color when color blinking occurs.</td>
</tr>
<tr>
<td>BlinkFrom</td>
<td>Double</td>
<td>0.0</td>
<td>The minimum value between 0.0 - 1.0 that will be sent to the property identified by BlinkTargetProperty.</td>
</tr>
<tr>
<td>BlinkTo</td>
<td>Double</td>
<td>1.0</td>
<td>The maximum value between 0.0 - 1.0 that will be sent to the property identified by BlinkTargetProperty.</td>
</tr>
<tr>
<td>BlinkBy</td>
<td>Double</td>
<td></td>
<td>The rate of change of the value sent to the property identified by BlinkTargetProperty.</td>
</tr>
<tr>
<td>BlinkInverse</td>
<td>Boolean</td>
<td>False</td>
<td>If set to True, blinking will be backward from the normal sense. This would allow the control to have two elements that blink in counterpoint.</td>
</tr>
<tr>
<td>BlinkSpeedRatio</td>
<td>Double</td>
<td>1.0</td>
<td>Specifies the rate of blinking, where 1.0 is normal speed, &lt;1.0 is slower and &gt;1.0 is faster than normal.</td>
</tr>
<tr>
<td>BlinkFillBehavior</td>
<td>One of: HoldEnd, Stop</td>
<td>Stop</td>
<td>Determines the state of the symbol when blinking</td>
</tr>
</tbody>
</table>

Type: Blink

Blink causes the symbol element to blink according to the condition or blink override settings.
Creating Custom Symbols

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlinkEasingFunction</td>
<td>One of: None,</td>
<td>None</td>
<td>Indicates the type of easing function to apply when approaching the limit of a blink based on fading. Some easing functions require parameters.</td>
</tr>
<tr>
<td></td>
<td>Back, Bounce,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Circle, Cubic,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Elastic, Expo-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nential, Power,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quadratic,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quartic, Quintic,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BlinkEasingMode</td>
<td>One of: EaseOut,</td>
<td></td>
<td>Determines whether the easing function will be applied at the end of the blink animation, the beginning, or both.</td>
</tr>
<tr>
<td></td>
<td>EaseIn, EaseIn-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BlinkEasingArg1</td>
<td>Integer</td>
<td>0</td>
<td>The first argument to the BlinkEasingFunction if applicable.</td>
</tr>
<tr>
<td>BlinkEasingArg2</td>
<td>Integer</td>
<td>0</td>
<td>The second argument to the BlinkEasingFunction if applicable.</td>
</tr>
</tbody>
</table>

**Type: Rotation**

Elements within the symbol may rotate clockwise or counter-clockwise according to the condition or rotation override settings.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotationType</td>
<td>One of:</td>
<td>None</td>
<td>If this attribute is &quot;Normal&quot; then the element can rotate.</td>
</tr>
<tr>
<td></td>
<td>None,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RotationFrom</td>
<td>Double</td>
<td>0</td>
<td>The starting angle of rotation, in degrees.</td>
</tr>
<tr>
<td>RotationTo</td>
<td>Double</td>
<td>360</td>
<td>The finishing angle of rotation, in degrees.</td>
</tr>
<tr>
<td>RotationBy</td>
<td>Double</td>
<td></td>
<td>The number of degrees to rotate by in each animation frame.</td>
</tr>
<tr>
<td>RotationInverse</td>
<td>Boolean</td>
<td>False</td>
<td>If False, rotation will be in the clockwise direction.</td>
</tr>
</tbody>
</table>
Creating Custom Symbols

### Property Table

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td></td>
<td></td>
<td>True, rotation is counterclockwise.</td>
</tr>
<tr>
<td><strong>RotationSpeedRatio</strong></td>
<td>Double</td>
<td>1.0</td>
<td>Specifies the rate of rotation, where 1.0 is normal speed, &lt;1.0 is slower and &gt;1.0 is faster than normal.</td>
</tr>
<tr>
<td><strong>RotationFillBehavior</strong></td>
<td>One of: HoldEnd, Stop</td>
<td>Stop</td>
<td>Determines the state of the symbol when rotation stops. If set to Stop, the control will be reset to its initial rotation. If set to HoldEnd, the control will be left as it was when rotation stopped.</td>
</tr>
<tr>
<td><strong>RotationEasingFunction</strong></td>
<td>One of: None, Back, Bounce, Circle, Cubic, Elastic, Exponential, Power, Quadratic, Quintic, Sine</td>
<td>None</td>
<td>Indicates the type of easing function to apply when approaching the limit of a rotation. Some easing functions require parameters.</td>
</tr>
<tr>
<td><strong>RotationEasingMode</strong></td>
<td>One of: EaseOut, EaseIn, EaseInOut</td>
<td></td>
<td>Determines whether the easing function will be applied at the end of the rotation animation, the beginning, or both.</td>
</tr>
<tr>
<td><strong>RotationEasingArg1</strong></td>
<td>Integer</td>
<td></td>
<td>The first argument to the RotationEasingFunction if applicable.</td>
</tr>
<tr>
<td><strong>RotationEasingArg2</strong></td>
<td>Integer</td>
<td></td>
<td>The second argument to the RotationEasingFunction if applicable.</td>
</tr>
</tbody>
</table>

### Type: Progress

Elements within the symbol may indicate progress in the range of 0.0 - 1.0 (representing 0 - 100%). Elements may also take on a progress color.
### Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProgressColorTargetProperty</td>
<td>String (a Property Name)</td>
<td>The name of an element property that will be set to the progress color.</td>
</tr>
</tbody>
</table>

### Conditional animation

The primary feature of the Symbol Host control is the ability to change the animation of the symbol based on an input condition. The Symbol Host defines 5 states, each of which can be individually configured for Color, Text, Blink and Rotation. All of the properties in the Condition, Blink and Rotation categories in the table above can be used to affect the behaviour of the symbol on a per-state basis.

### Scaling and Other Considerations

#### Scaling

Your symbol will be resized when the Symbol Host control changes size. You will need to decide what kind of resizing behaviour you would like. Normally, resizing the control will simply scale the contents in the X and Y direction independently, allowing the user to stretch your symbol. This is not always the correct behaviour. Typical requirements for alternate scaling methods include:

1. If the symbol needs to maintain an aspect ratio, resizing should not cause flattening of the symbol, or
2. If the symbol contains controls such as text boxes or buttons then resizing should also cause the text and other contained controls to scale, or
3. If the symbol contains elements that have a fixed size, and therefore do not scale as the rest of the symbol scales.

Scaling issues can commonly be handled using `<Grid>` tags, which automatically resize all of their contained elements. When `<Grid>` tags are not sufficient, the best choice is to surround the entire symbol definition in a `<Viewbox>`:

```xml
<ControlTemplate x:Key="EllipseTemplate" TargetType="sym:DHSymbol">
    <Viewbox Stretch="Fill">
        <Grid Width="100" Height="100">
            <Ellipse sym:DHSymbol.BaseColorTargetProperty="Fill.Color" Fill="Transparent" Stroke="Black"/>
            <Rectangle Fill="Red" Stroke="Transparent" Width="20" sym:DHSymbol.BlinkTargetProperty="Opacity" sym:DHSymbol.RotationType="Normal"/>
        </Grid>
    </Viewbox>
</ControlTemplate>
```
Notice that the content within the Viewbox must include a specific width and height. The Viewbox must know the nominal size of its contained element, and will scale the content relative to that size.

**Binding element properties to the symbol host control**

The Symbol Host control is an instance of the class DHSymbol. Since the symbol content is specified as a ControlTemplate, any of the publicly exposed members of the host control can be accessed by the symbol XAML using `{TemplateBinding property_path}`. This allows the symbol designer to reflect the host control state within the symbol if necessary. For example, the symbol could contain a text box that presents the progress percentage as a number.

It is not always a good idea to use TemplateBinding instead of the DHSymbol attached properties. The attached properties act as markup that is not specific to a particular Symbol Host control. If you chose to implement your own symbol host control or use a third-party symbol host control then the attached property markup would make your symbol relatively easy to support. If you choose instead to bind your symbol using TemplateBinding then your symbol will be much less likely to function with another symbol host control.

**Limitations on XAML code within symbols**

A symbol can contain virtually any XAML code with the exception of event handlers. Event handlers require C# code-behind, which is not available to a symbol. Fairly complex behaviours can be achieved with storyboards and bindings among elements in the XAML definition.

**Compressing symbol files**

The XAML file containing the symbol definitions can be compressed with the standard ZIP compression. This will reduce loading time and network bandwidth when the application starts. You may not specify a password on this ZIP file.

**Best Practices**

When symbols are intended to line up with one another, use integer dimensions and positions. Stay consistent when offsetting symbol contents from the control boundary. For example, a symbol set consisting of pumps and pipes would be much easier to work with if the pipe were easily scaled to match the size of the pump's output, and if the pipes were offset within their bounding box by the size of the flange on the pump's output.

Create symbols that consist of multiple layers. The bottom layer should be filled with the base color, the second layer with the condition color and the top layer with a translucent overlay as a lighting effect whose opacity is tied to the glassiness. ZIP your XAML files when you deploy them to save time and network bandwidth.
Complete Example

The following two listings provide a complete example that implements 4 symbols:

- Spinner: A circle containing a rotating rectangle.
- Spiral: A spiral path
- Info (Attached): A rectangle containing symbol information using attached properties to create its bindings.
- Info (Template): The same rectangle using template bindings.

File: TestSymbolsV1.0.map

```xml
<?xml version="1.0" encoding="utf-8"?>
<wvsm:SymbolMap
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wvsm="clr-namespace:Cogent.DataHubWebView;assembly=DataHubWebViewSymbolHelpers"
    SymbolSet="My Test Symbols"
    Manufacturer="Cogent Real-Time Systems Inc."
    Copyright="(C) 2011 Cogent Real-Time Systems Inc."
    License="Licensed for use with DataHub WebView only."
    Description="These Symbols illustrate how to incorporate custom objects into DataHub WebView."
    SymbolSetVersion="1.0"
>
    <wvsm:SymbolMap.Symbols>
        <wvsm:Symbol Category="Simple Shapes" Name="Spinner" SymbolID="1"
            FileName="SimpleShapes.xaml" Style="SpinnerStyle" />
        <wvsm:Symbol Category="Simple Shapes" Name="Spiral" SymbolID="2"
            FileName="SimpleShapes.xaml" Style="SpiralStyle" />
        <wvsm:Symbol Category="Simple Shapes" Name="Info (Template)"
            SymbolID="3" FileName="SimpleShapes.xaml" Style="Info1Style" />
        <wvsm:Symbol Category="Simple Shapes" Name="Info (Attached)"
            SymbolID="4" FileName="SimpleShapes.xaml" Style="Info2Style" />
    </wvsm:SymbolMap.Symbols>
</wvsm:SymbolMap>
```

(Long strings have been wrapped [56])

File: SimpleShapes.xaml
<UserControl
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
mc:Ignorable="d"
xmlns:sym=
"clr-namepsace:Cogent.DataHubWebView.Controls;assembly=
DataHubWebViewSymbol"
xmlns:wvc=
"clr-namepsace:Cogent.DataHubWebView.Controls;assembly=
DataHubWebViewControlsNew"
>
<UserControl.Resources>

<!-- This string converter is used with {Binding} definitions to convert values into strings where they are then submitted to another control property and converted to the appropriate type as part of the XAML parsing. This is a quick way to perform a type conversion that isn't naturally supported, such as Color to Brush. -->
<wvc:ObjectToStringConverter x:Key="stringConverter" />

<ControlTemplate x:Key="SpinnerTemplate" TargetType="sym:DHSymbol">
<Grid Width="100" Height="100">
  <Ellipse sym:DHSymbol.BaseColorTargetProperty="Fill.Color"
    Fill="White" Stroke="Black"/>
  <Rectangle Fill="Red" Stroke="Transparent" Width="10"
    sym:DHSymbol.BlinkTargetProperty="Opacity"
    sym:DHSymbol.RotationType="Normal"/>
</Grid>
</Viewbox>
</ControlTemplate>

<ControlTemplate x:Key="SpiralTemplate" TargetType="sym:DHSymbol">
<Viewbox Stretch="Uniform" sym:DHSymbol.RotationType="Normal">
  <Path sym:DHSymbol.AccentColor1TargetProperty="Stroke.Color"
    sym:DHSymbol.BaseColorTargetProperty="Fill.Color"
    Fill="Transparent" StrokeThickness="2"
    Data="M45,50 A5,5 180 1 1 55,50 M40,50 A7,7 -180 1 0 55,50"/>
</Viewbox>
</ControlTemplate>
</UserControl>
Creating Custom Symbols

```xml
<Path Stroke>
  <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
    <GradientStop Offset="0.0" Color="Green"/>
    <GradientStop Offset="0.0" Color="Green"/>
    <GradientStop Offset="1.0" Color="Yellow"/>
    <GradientStop Offset="1.0" Color="Yellow"/>
  </LinearGradientBrush>
</Path.Stroke>
</Path>
</Grid>
</Viewbox>
</ControlTemplate>

<!-- Create a rectangular information block using template binding. -->
<ControlTemplate x:Key="InfoTemplate1" TargetType="sym:DHSymbol">
  <Grid>
    <Grid.Resources>
      <SolidColorBrush x:Key="SymbolBaseColor" Color="{Binding RelativeSource={RelativeSource TemplatedParent}, Path=BaseColor}"
      <LinearGradientBrush x:Key="HorizontalAccentGradient" StartPoint="0 0" EndPoint="1 0">
        <GradientStop Offset="0.0" Color="{Binding RelativeSource={RelativeSource TemplatedParent}, Path=AccentColor1}"
          <GradientStop Offset="1.0" Color="{Binding RelativeSource={RelativeSource TemplatedParent}, Path=AccentColor2}"
      </LinearGradientBrush>
      <LinearGradientBrush x:Key="VerticalProgressGradient" StartPoint="0 1" EndPoint="0 0">
        <GradientStop Offset="0.0" Color="{Binding RelativeSource={RelativeSource TemplatedParent}, Path=ProgressColor}"
        <GradientStop Offset="1.0" Color="{Binding RelativeSource={RelativeSource TemplatedParent}, Path=ProgressColor}"
      </LinearGradientBrush>
    </Grid.Resources>
```

M40,50 A10,10 180 1 1 60,50 M35,50 A12,12 -180 1 0 60,50
M35,50 A15,15 180 1 1 65,50 M30,50 A17,17 -180 1 0 65,50
M30,50 A20,20 180 1 1 70,50 M25,50 A22,22 -180 1 0 70,50
M25,50 A25,25 180 1 1 75,50 M20,50 A27,27 -180 1 0 75,50
M20,50 A30,30 180 1 1 80,50 M15,50 A32,32 -180 1 0 80,50
M15,50 A35,35 180 1 1 85,50 M10,50 A37,37 -180 1 0 85,50
M10,50 A40,40 180 1 1 90,50 M5,50 A42,42 -180 1 0 90,50
M5,50 A45,45 180 1 1 95,50 M0,50 A47,47 -180 1 0 95,50"
Path=ProgressPercentage}"
    Color="{(Binding RelativeSource={RelativeSource TemplatedParent},
    Path=ProgressColor}"
    <GradientStop Offset=""
        "{(Binding RelativeSource={RelativeSource TemplatedParent},
    Path=ProgressPercentage}" Color="Transparent"/>
    <GradientStop Offset="1.0" Color="Transparent"/>
    </LinearGradientBrush>
</Grid.Resources>
<!-- When we use template binding instead of attached properties, WebView cannot tell that the properties are in use, so it will not correctly display supported behaviours in its property editor. We create an invisible control that attaches the properties that we want to show up as supported in the property editor. -->
<Border Background="Transparent"
    sym:DHSymbol.AccentColor1TargetProperty="Background.Color"
    sym:DHSymbol.AccentColor2TargetProperty="Background.Color"
    sym:DHSymbol.BaseColorTargetProperty="Background.Color"
    sym:DHSymbol.ProgressColorTargetProperty="Background.Color"
    sym:DHSymbol.ConditionColorTargetProperty="Background.Color"
    sym:DHSymbol.GlassinessTargetProperty="Opacity"
    sym:DHSymbol.ConditionColorOpacityTargetProperty="Opacity"
    sym:DHSymbol.RotationType="Normal"
    Visibility="Collapsed">
    <TextBlock Text=""
        sym:DHSymbol.ConditionTextTargetProperty="Text"
        sym:DHSymbol.ProgressPercentageTargetProperty="Text"/>
</Border>
<!-- Create a rectangle. TemplateBinding does not support a Converter, but Binding does. -->
<Rectangle Opacity="{(TemplateBinding Glassiness}" Stroke="Black"
    Fill="{(StaticResource VerticalProgressGradient})"/>
<!-- Create some other controls inside the rectangle that bind to other properties of the symbol host. -->
<StackPanel Margin="1,1,1,1"
    Opacity="{(TemplateBinding ConditionColorOpacity})"
    <CheckBox Content="{(TemplateBinding ConditionText)"
        IsChecked="{(TemplateBinding IsRotating)" IsEnabled="False"/>
    <StackPanel Orientation="Horizontal"
        Background="{(StaticResource HorizontalAccentGradient})"
    <TextBlock Text="Progress: " Margin="0,0,5,0"
        Foreground="{(StaticResource SymbolBaseColor}"/>
Creating Custom Symbols

<!-- Create a rectangular information block using attached properties. -->
<ControlTemplate x:Key="InfoTemplate2" TargetType="sym:DHSymbol">
  <Grid>
    <Grid.Resources>
      <LinearGradientBrush x:Key="HorizontalAccentGradient"
       StartPoint="0 0" EndPoint="1 0">
        <GradientStop Offset="0.0" Color="White"/>
        <GradientStop Offset="1.0" Color="Black"/>
      </LinearGradientBrush>
      <LinearGradientBrush x:Key="VerticalProgressGradient"
       StartPoint="0 1" EndPoint="0 0">
        <GradientStop Offset="0.0" Color="White"/>
        <GradientStop Offset="0.0" Color="White"/>
        <GradientStop Offset="1.0" Color="Transparent"/>
        <GradientStop Offset="1.0" Color="Transparent"/>
      </LinearGradientBrush>
    </Grid.Resources>
    <Rectangle sym:DHSymbol.GlassinessTargetProperty="Opacity"
     Stroke="Black"
     Fill="{StaticResource VerticalProgressGradient}"
     sym:DHSymbol.ProgressPercentageTargetProperty=
     "Fill.(LinearGradientBrush.GradientStops)[1].Offset,Fill.(LinearGradientBrush.GradientStops)[2].Offset"
    />
  </Grid>
</ControlTemplate>

<!-- Create the same rectangular information block using template binding. -->
<ControlTemplate x:Key="InfoTemplate1" TargetType="sym:DHSymbol">
  <Grid>
    <StackPanel>
      <TextBlock Foreground="{StaticResource SymbolBaseColor}" Text="{Binding RelativeSource={RelativeSource TemplatedParent}, Path=ProgressPercentage, Converter={StaticResource stringConverter}}"/>
    </StackPanel>
  </Grid>
</ControlTemplate>

<!-- Create some other controls inside the rectangle that bind to other properties of the symbol host. Note that the condition text will only show up if the "Show Condition -->
Creating Custom Symbols

Text on Symbol checkbox is checked in the property category "Advanced Configuration: Condition Color and Text". -->

<StackPanel Margin="1,1,1,1"
    sym:DHSymbol.ConditionColorOpacityTargetProperty="Opacity"/>

<!-- We have no way to reach IsRotating through an attached property. We must use TemplateBinding for that. -->

<CheckBox IsChecked="{TemplateBinding IsRotating}" IsEnabled="False"
    sym:DHSymbol.ConditionTextTargetProperty="Content"/>

<StackPanel Orientation="Horizontal"
    Background="{StaticResource HorizontalAccentGradient}"
    sym:DHSymbol.AccentColor1TargetProperty="Background.(LinearGradientBrush.GradientStops)[0].Color"
    sym:DHSymbol.AccentColor2TargetProperty="Background.(LinearGradientBrush.GradientStops)[1].Color"
>
    <TextBlock Text="Progress: "
        Margin="0,0,5,0" sym:DHSymbol.
        BaseColorTargetProperty="Foreground.Color"
        Foreground="Black"
        sym:DHSymbol.BaseColorTargetProperty="Foreground.Color"
        Foreground="Black"

        <TextBlock sym:DHSymbol.ProgressPercentageTargetProperty="Text"/>

    </StackPanel>
</StackPanel>
Creating Custom Symbols

```xml
<Setter Property="Template"
      Value="{StaticResource InfoTemplate1}" />
</Style>

<Style x:Key="Info2Style" TargetType="sym:DHSymbol">
  <Setter Property="StyleName" Value="InfoAttached" />
  <Setter Property="Template"
      Value="{StaticResource InfoTemplate2}" />
</Style>

</UserControl.Resources>

<sym:DHSymbol Style="{StaticResource SpinnerStyle}" />
</UserControl>

(Long strings have been wrapped [56])
WebView Controls
Advanced Check Box

Advanced Check Box — toggles between two states, each with advanced properties.

Description

Changes between two states: checked and unchecked. Each state has additional user properties: a numeric value, a string, a DateTime, and a color.
Alarm List

Alarm List — a table that displays alarms and events.

Description

The Alarm control lets you view the most recent alarms at the top and events at the bottom. The alarms data comes from a data domain in the Cogent DataHub that has been configured for OPC Alarms and Events (OPC A&E).

Use

The Select Columns button for each section lets you choose which columns to display. The Default Sort button for each section reverts the sort order back to the default setting. The Ack button allows you to acknowledge and remove all the alarms, while the buttons in that column let you acknowledge and remove individual alarms.
Boolean Converter

Boolean Converter — a program block that selects between two states.

Description

Selects an output value, an output DateTime and an output color based on an input boolean value (True/False). This is a design-time control and is not visible in Run mode.
Calendar

Calendar — displays a calendar.

Description

Displays a calendar, enabling the user to select a date.
Circular Gauge 1

Circular Gauge 1 — simple circular gauge.

Description

Used to graphically represent real-time data. In its default configuration, this gauge is a simple circle with a needle. With Edit mode set to Drag, the user can interact with the gauge in Run mode. This is a highly-configurable control. Properties like value, value range, editing ability, and angle range are all modifiable.
Circular Gauge 2

Circular Gauge 2 — circular gauge with varying value ranges and an indicator light.

Description

Used to graphically represent real-time data. This gauge has an indicator light that shows whether the needle is in the optimal range, below optimal, or above optimal. This gauge does not allow user interaction in Run mode. All ranges, sizes, values, and colors can be changed.
Color Selector

Color Selector — a palette used to store specific colors and to access application theme colors.

Description

Used to set a variety of colors as Custom Colors and to access application Theme Colors. These colors can then be used as binding sources for other controls on the page. This makes it easy to create and maintain custom color themes. This is a design-time control and is not visible in Run mode.
Color Selector

Color Selector — a program block that uses ARGB values to produce a color.

Description

Uses four numeric inputs (alpha, red, green, blue), each in the range 0-255, to build a color. Alternatively, references a color from a Color Palette. This is a design-time control and is not visible in Run mode.
**ComboBox**

ComboBox — a simple dropdown list used for item selection.

**Description**

Enables the user to select from among available items. The list of items can be configured as a comma-separated list, or bound to the result of an expression.
Comparator

Comparator — a program block that compares two values and outputs the results.

Description

Compares two inputs (DataPoints or values) and produces various output values.optionally, accepts a numeric tolerance to stabilize comparisons based on rapidly-changing inputs. This is a design-time control and is not visible in Run mode.
Condition Selector

Condition Selector — a program block used to select among five different states.

Description

Selects output text and values based on a condition input value. The input can be treated as a boolean to produce a two-state result, or can be compared to each state's value range. If states have overlapping ranges, the first match is used. Each state can be associated with text, colors and values. Typically, these configurable state settings feed other controls and processes. This is a design-time control and is not visible in Run mode.
Control Panel

Control Panel — supports Run mode option changes.

Description

Enables the user to change various options while in Run mode.
Date Picker

Date Picker — allows a user to select a date.

Description

Displays an entry field that opens a calendar, enabling the user to select a date.

Format Strings

Numeric values, text, dates, and times can be formatted using Windows standard and custom format strings. For example, to format a numeric value to two decimal places, specify a format of 0.00. For more information on formatting numbers, text, dates and times, please see the section called “Format Strings”.
Filtered Data Table

Filtered Data Table — row/column results from a database query.

Description

Presents the result of a row/column data set in a table. Columns can be reordered and pinned. Rows can be filtered and grouped.
**Hi/Low Indicator**

Hi/Low Indicator — changes color to indicate high and low values.

**Description**

Responds to real-time data updates by changing color. The color corresponds to a matching value range. Five ranges can be configured: Low Low, Low, Normal, High, and High High. It has modifiable text, limits, colors, and transition time.

**Format Strings**

Numeric values, text, dates, and times can be formatted using Windows standard and custom format strings. For example, to format a numeric value to two decimal places, specify a format of `0.00`. For more information on formatting numbers, text, dates and times, please see the section called “Format Strings”. 
**Horizontal Linear Gauge**

Horizontal Linear Gauge — a linear horizontal gauge with a slider,

**Description**

Used to graphically represent real-time data. In its default configuration, this linear gauge is oriented horizontally. With Edit mode set to Drag, the user can interact with the gauge in Run mode. This is a highly-configurable control. Properties like the value, value range, editing ability, and angle range are all modifiable.
Hyperlink Button

Hyperlink Button — acts as a hyperlink to another page or a URL.

Description

Uses a button to link the user to another page or to an external URL. This control is often used to provide navigation support among a collection of related pages.
Hyperlink Image

Hyperlink Image — acts as a hyperlink to another page or a URL.

Description

Uses an image to link the user to another WebView page or to an external URL. This control is often used to provide navigation support among a collection of related pages.
Hyperlink Text

Hyperlink Text — acts as a hyperlink to another page or a URL.

Description

Uses a text label to link the user to another page or to an external URL. This control is often used to provide navigation support among a collection of related pages.
Image

Image — an image file container.

Description

Displays an image located in the images file directory on the web server.
**Left 90 Degree Gauge**

Left 90 Degree Gauge — a quarter-circle gauge.

**Description**

Used to graphically represent real-time data. In its default configuration, this gauge is a quarter-circle with a needle. With Edit mode set to Drag, the user can interact with the gauge in Run mode. This is a highly-configurable control. Properties like the value, value range, editing ability, and angle range are all modifiable.
List Box

List Box — a simple list used for item selection.

Description

Enables the user to select from among available items. The list of items can be configured as a comma-separated list, or bound to the result of an expression.
Media Player

Media Player — plays audio and videos.

Description

Displays media located in the media file directory on the web server. Typical playback controls are available. Media can be configured to auto play and auto repeat. Audio controls are also available.
One Input Calculator

One Input Calculator — a program block that performs calculations on a single input value.

Description

Calculates a variety of values based on a single input. Calculations include Boolean and Duration conversions, mathematical operations (Abs, Sign, Ceiling, Floor, Exponent, Log, Square, SquareRoot), and trigonometric functions (Sin, Cos, Tan, etc). This is a design-time control and is not visible in Run mode.
Point Data Table

Point Data Table — a table consisting of all available data points.

Description

Presents all available data points in the DataHub in a table format. The columns available are Point Name, Display Name, Value, Timestamp, Quality Name, and Quality.
Polynomial Calculator

Polynomial Calculator — a program block that calculates the result of a polynomial expression.

Description

Calculates the result of a polynomial expression (up to 5th order). This is a design-time control and is not visible in Run mode.
Progress Bar

Progress Bar — an expanding/shrinking progress bar.

Description

Used to graphically represent real-time data. This control shows the input value by expanding or shrinking in size. The bar's orientation, range, and size can all be modified. The progress bar also accommodates color change based on five ranges: Low Low, Low, Normal, High, and High High.
Radio Button

Radio Button — a button that offers a choice of mutually exclusive options.

Description

Used in groups where only one of the buttons in the group can be checked at a time (i.e., mutually exclusive). The Control Value for each radio button in a group should be bound to a single source (e.g., data point), which will be treated as the group's input. The radio button whose value matches the Control Value will be selected.

Your text here
Range Mapper

Range Mapper — a program block that maps an input to an output, using ranges.

Description

Maps a single input (DataPoint or value) to a value in a corresponding output range. Ranges are specified with an input minimum and maximum, and an output minimum and maximum. The input can be clamped for limited, linear interpolation, or unclamped for extrapolation. This is a design-time control and is not visible in Run mode.
**Rising/Falling Indicator**

Rising/Falling Indicator — a display that changes according to the rise or fall of a value.

**Description**

Responds to real-time data updates by changing color. The color reflects how quickly the input value is rising or falling. It has modifiable text, colors, transition time, and steady time.

**Format Strings**

Numeric values, text, dates, and times can be formatted using Windows standard and custom format strings. For example, to format a numeric value to two decimal places, specify a format of `0.00`. For more information on formatting numbers, text, dates and times, please see the section called “Format Strings”.
Semi-circular Gauge

Description

Used to graphically represent real-time data. In its default configuration, this gauge is a semi-circle with a needle. With Edit Mode set to Drag, the user can interact with the gauge in Run mode. This is a highly-configurable control. Properties like the value, value range, editing ability, and angle range are all modifiable.
Series Chart

Series Chart — displays data in chart format - bars, lines, pie, etc.

Description

Provides a number of different chart formats for displaying related data values.

The Series Chart control requires tabular data, where the first column of the table is the X axis and all other columns are the Y axis data. The control automatically treats all data columns after the first one as independent series sharing the same X axis. You need to construct such a table in a database or as an Excel range, and write it to a single data point in the DataHub. That point can then be bound to the Series Chart control. It is possible to display multiple series charts.
Shining Light

Shining Light — an indicator light that can flash and change color.

Description

Displays a light which responds to boolean triggers and color changes. This control is typically used for notification. Boolean inputs control whether the light is on or flashing. The light color can be set with a single input, or configured using gradient colors and offsets. Duration, auto reverse, and repeat behavior are also modifiable.
Simple Button

Simple Button — a simple, clickable button.

Description

Provides a simple way to attach script code to a button click event.
**Simple Check Box**

Simple Check Box — toggles between two states.

**Description**

Changes between two states: checked and unchecked. Typically, a DataPoint is bound to the input value and used to toggle between states. Each state has an associated value which can be used to feed another control or process.
Simple Ellipse

Simple Ellipse — a simple ellipse with editable appearance and properties.

Description

A simple ellipse with modifiable fill color, stroke color, and stroke thickness.
**Simple Path**

Simple Path — a path that can create any shape.

**Description**

A simple path that can be used to create any shape through mapping out each point using XAML path notation. Fill color, stroke color, canvas size, stroke thickness, stroke caps, stroke joins, and stroke miter limit are all modifiable. Please see the XAML path notation for [Path Markup Syntax](#) in the online Microsoft reference library for more information.
Simple Radial Gauge

Simple Radial Gauge — a simple, tutorial-oriented circular gauge.

Description

A simple, tutorial-oriented circular gauge that illustrates how to build custom controls. This customizable radial gauge has a scale and a needle. Developed using RadControls by Telerik Corporation.
Simple Rectangle

Simple Rectangle — a simple rectangle with editable appearance and properties.

Description

A simple rectangle with modifiable fill color, stroke color, stroke thickness and corner radius.
Slider

Slider — a scale with an adjustable slider to control or view values.

Description

Enables the user to select a value along a configurable scale by dragging the slider. The slider can be configured either horizontally or vertically. Axis labels can be displayed before or after the scale.
Symbol

Symbol — a container for over 4000 symbols.

Description

A common container for over 4000 industry standard symbols. Symbols can be configured to blink, rotate and show progress. Output states are selected based on a condition input value. The input can be treated as a boolean to produce a two-state result, or can be compared to each state's value range. If states have overlapping ranges, the first match is used. Each state is associated with a value range, color, text, blinking, blink rate, rotating, and rotation rate. Symbols can be automatically updated when new versions of the symbol are available.
System Information

System Information — a program block that can access system, user, and page information.

Description

Accesses information about the system and page. Outputs include local time, user, page name, file name, description, and owner. Typically, these values are used for page titles and footers. This is a design-time control and is not visible in Run mode.
Text Entry Field

Text Entry Field — a simple textbox used for data entry.

Description

Enables the user to input a text string or numeric value while in Run mode.

Format Strings

Numeric values, text, dates, and times can be formatted using Windows standard and custom format strings. For example, to format a numeric value to two decimal places, specify a format of `0.00`. For more information on formatting numbers, text, dates and times, please see the section called “Format Strings”.

Text Label

Text Label — a textual display with no entry field.

Description

A text label that displays text, but does not have an entry field. Color is modifiable.

Format Strings

Numeric values, text, dates, and times can be formatted using Windows standard and custom format strings. For example, to format a numeric value to two decimal places, specify a format of 0.00. For more information on formatting numbers, text, dates and times, please see the section called “Format Strings”.
Thermometer

Thermometer — a tutorial-oriented linear gauge.

Description

A tutorial-oriented linear gauge that illustrates how to build custom controls. This highly customizable linear gauge has two linear scales, adjustable offsets and the ability to measure in either Celsius or Fahrenheit. Developed using RadControls by Telerik Corporation.
Three Indicator Radial Gauge

Three Indicator Radial Gauge — a tutorial-oriented, multi-indicator, circular gauge.

Description

A tutorial-oriented, multi-indicator, circular gauge that illustrates how to build custom controls. This highly customizable radial gauge has one scale, three indicators, and a three-part range list. Developed using RadControls by Telerik Corporation.
Three Point Slider

Three Point Slider — a horizontal gauge that shows up to three data points on a slider.

Description

Used to graphically represent real-time data. This slider shows up to three values and has optimal, below optimal, and above optimal ranges. It has a slider for the primary value, which the user can drag in Run mode, and a progress bar for each of the other values. It also has an error indicator which flashes when the primary value is outside the optimal range.
Time Picker

Time Picker — allows a user to select a time.

Description

Displays an entry field that enables the user to select a time.

Format Strings

Numeric values, text, dates, and times can be formatted using Windows standard and custom format strings. For example, to format a numeric value to two decimal places, specify a format of 0.00. For more information on formatting numbers, text, dates and times, please see the section called “Format Strings”.

Timer

Timer — a program block executes that provides timer and counter behavior.

Description

Provides timer behavior to control process execution. Counter functions include Increment, Decrement and Toggle. Supports common repeat behavior. This is a design-time control and is not visible in Run mode.
## Toggle Button

Toggle Button — a push button with an optional two-state toggle.

### Description

Has a normal and a pushed state that can have toggle behavior or a press-and-release behavior. Each state has associated text, text color, and a numeric value.
Top Sweep Gauge

Top Sweep Gauge — a horizontal curved gauge.

Description

Used to graphically represent real-time data. In its default configuration, this gauge is a curved upper-portion of a circle with a needle. With Edit mode set to Drag, the user can interact with the gauge in Run mode. This is a highly-configurable control. Properties like the value, value range, editing ability, and angle range are all modifiable.
Trend

Trend — two chart controls can track up to 3 or 8 data points.

Description

These two charts (3-pen and 8-pen) allow a user to assign values and data points to three or eight trend lines. They track the values and variations of each point as they change over time. These Trend Charts also leverage the power of the Data Historian.
Two Input Calculator

Two Input Calculator — a program block that performs calculations on two input values.

Description

Calculates various mathematical and logical output values using two inputs (DataPoints or values). Functions include: Sum, Difference, Product, Quotient, Modulo, Minimum, Maximum, Round, etc. This is a design-time control and is not visible in Run mode.
Vertical Linear Gauge

Vertically Linear Gauge — a linear vertical gauge with a slider.

Description

Used to graphically represent real-time data. In its default configuration, this linear gauge is oriented vertically. With Edit mode set to Drag, the user can interact with the gauge in Run mode. This is a highly-configurable control. Properties like the value, value range, editing ability, and angle range are all modifiable.
# Table of Contents

**Introduction** .................................................................................................................... 1

**What’s Different About DataHub Scripting?** ..................................................................... 2

**Scripts and their Environment** .......................................................................................... 2

  - Dynamic Environment ..................................................................................................... 2
  - Event Driven .................................................................................................................. 2
  - Object Oriented .............................................................................................................. 3

**Symbols, Variables, and Evaluation** ................................................................................. 5

  - Symbols and Variables ................................................................................................. 5
  - The Read/Evaluate Cycle .............................................................................................. 5

**Access to DataHub Points** ................................................................................................. 8

  - Point Names .................................................................................................................. 8
  - Point Values .................................................................................................................... 9
  - Point Timestamps and Qualities .................................................................................... 9

**ODBC and Windows Scripting** .......................................................................................... 9

  - DataHub ODBC (Open Database Connectivity) Scripting .............................................. 9
  - DataHub Windows Scripting ......................................................................................... 10

**Getting Started** .................................................................................................................. 11

  - How to Run a Script ..................................................................................................... 11
  - The Script Editor ........................................................................................................... 12
  - The Script Log ............................................................................................................... 13
  - The Script Application Manager ................................................................................... 15

**Writing Scripts** .................................................................................................................... 16

  - Creating a Script .......................................................................................................... 16
  - Hello World .................................................................................................................... 18
  - Accessing Data .............................................................................................................. 20
  - Modifying Data .............................................................................................................. 22
  - Making a Window .......................................................................................................... 23
  - Encrypting a Script ....................................................................................................... 25
  - Scripting Tips ................................................................................................................ 26
    - Copying a complete tutorial .................................................................................... 26
    - Setting up a scripting environment ........................................................................ 27

**The Application class** ....................................................................................................... 28

  - Class Definition ........................................................................................................... 28
  - Construction and Destruction ...................................................................................... 29
  - Handling Events ............................................................................................................ 29
  - Timers ............................................................................................................................. 30
  - Menus .............................................................................................................................. 31

**Example Scripts** ................................................................................................................ 36

  - LogFile.g ....................................................................................................................... 37
  - ReadCSV.g ...................................................................................................................... 40
  - WriteCSV.g .................................................................................................................... 47
  - XMLReader.g .................................................................................................................. 53
  - ParseExcel.g ................................................................................................................... 56
  - LinearXform.g ............................................................................................................... 62
Methods and Functions from Special Gamma Functions for DataHub Scripting ........................................................ 106

Built-in Classes .................................................................................................................................................. 102

Special Gamma Functions for DataHub Scripting ......................................................................................... 106

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DH_Item</td>
<td>104</td>
</tr>
<tr>
<td>DH_Domain</td>
<td>103</td>
</tr>
<tr>
<td>AutoCalculation.g</td>
<td>98</td>
</tr>
<tr>
<td>OPCReconnect.g</td>
<td>94</td>
</tr>
<tr>
<td>OPCItemLoader.g</td>
<td>90</td>
</tr>
<tr>
<td>OPCReload.g</td>
<td>87</td>
</tr>
<tr>
<td>FixQuality.g</td>
<td>87</td>
</tr>
<tr>
<td>set_point_flush_flags</td>
<td>127</td>
</tr>
<tr>
<td>on_change</td>
<td>124</td>
</tr>
<tr>
<td>set_point_queue_depth</td>
<td>123</td>
</tr>
<tr>
<td>datahub_log</td>
<td>125</td>
</tr>
<tr>
<td>get_point_queue_count</td>
<td>122</td>
</tr>
<tr>
<td>datahub_domaininfo</td>
<td>113</td>
</tr>
<tr>
<td>get_point_queue_depth</td>
<td>123</td>
</tr>
<tr>
<td>datahub_domains</td>
<td>115</td>
</tr>
<tr>
<td>datahub_log</td>
<td>116</td>
</tr>
<tr>
<td>datahub_domaininfo</td>
<td>117</td>
</tr>
<tr>
<td>datahub_read</td>
<td>119</td>
</tr>
<tr>
<td>datahub_write</td>
<td>120</td>
</tr>
<tr>
<td>edit_file</td>
<td>121</td>
</tr>
<tr>
<td>datahub_command</td>
<td>111</td>
</tr>
<tr>
<td>get_tray_menu</td>
<td>124</td>
</tr>
<tr>
<td>allow_self_reference</td>
<td>110</td>
</tr>
<tr>
<td>remove_menu_action</td>
<td>127</td>
</tr>
<tr>
<td>remove_change</td>
<td>125</td>
</tr>
<tr>
<td>set_point_queue_depth</td>
<td>123</td>
</tr>
<tr>
<td>set_point_flush_flags</td>
<td>128</td>
</tr>
<tr>
<td>show_log</td>
<td>131</td>
</tr>
<tr>
<td>symcmp</td>
<td>132</td>
</tr>
</tbody>
</table>

Methods and Functions from Application.g ......................................................................................... 133

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSubMenu</td>
<td>139</td>
</tr>
<tr>
<td>AddPermanentMenuItem</td>
<td>137</td>
</tr>
<tr>
<td>AddStopMenuItem</td>
<td>139</td>
</tr>
<tr>
<td>AddStartMenuItem</td>
<td>138</td>
</tr>
<tr>
<td>AddMenuItem</td>
<td>136</td>
</tr>
<tr>
<td>AddCustomSubMenu</td>
<td>135</td>
</tr>
<tr>
<td>AddCustomMenuItem</td>
<td>134</td>
</tr>
<tr>
<td>AddmenuItem</td>
<td>108</td>
</tr>
<tr>
<td>allow_self_reference</td>
<td>110</td>
</tr>
<tr>
<td>datahub_command</td>
<td>111</td>
</tr>
<tr>
<td>datahub_domaininfo</td>
<td>113</td>
</tr>
<tr>
<td>datahub_domains</td>
<td>115</td>
</tr>
<tr>
<td>datahub_log</td>
<td>116</td>
</tr>
<tr>
<td>datahub_point</td>
<td>117</td>
</tr>
<tr>
<td>datahub_read</td>
<td>119</td>
</tr>
<tr>
<td>datahub_write</td>
<td>120</td>
</tr>
<tr>
<td>edit_file</td>
<td>121</td>
</tr>
<tr>
<td>get_point_queue_depth</td>
<td>123</td>
</tr>
<tr>
<td>on_change</td>
<td>124</td>
</tr>
<tr>
<td>remove_menu_action</td>
<td>127</td>
</tr>
<tr>
<td>remove_change</td>
<td>125</td>
</tr>
<tr>
<td>set_point_queue_depth</td>
<td>123</td>
</tr>
<tr>
<td>set_point_flush_flags</td>
<td>128</td>
</tr>
<tr>
<td>show_log</td>
<td>131</td>
</tr>
<tr>
<td>symcmp</td>
<td>132</td>
</tr>
</tbody>
</table>

Methods and Functions from Application.g ......................................................................................... 133
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationMultiple</td>
<td>141</td>
</tr>
<tr>
<td>ApplicationSingleton</td>
<td>142</td>
</tr>
<tr>
<td>CreateSystemMenu</td>
<td>143</td>
</tr>
<tr>
<td>droptimer</td>
<td>144</td>
</tr>
<tr>
<td>OnChange</td>
<td>145</td>
</tr>
<tr>
<td>RemoveAllChanges</td>
<td>146</td>
</tr>
<tr>
<td>RemoveAllEventHandlers</td>
<td>147</td>
</tr>
<tr>
<td>RemoveAllMenus</td>
<td>148</td>
</tr>
<tr>
<td>RemoveAllTimers</td>
<td>149</td>
</tr>
<tr>
<td>RemoveChange</td>
<td>150</td>
</tr>
<tr>
<td>RemoveSystemMenu</td>
<td>151</td>
</tr>
<tr>
<td>RemoveTimer</td>
<td>152</td>
</tr>
<tr>
<td>TimerAt</td>
<td>153</td>
</tr>
<tr>
<td>TimerEvery</td>
<td>154</td>
</tr>
<tr>
<td>TimerEvery</td>
<td>156</td>
</tr>
<tr>
<td>Time Conversion Functions from Time.g</td>
<td>157</td>
</tr>
<tr>
<td>GetCurrentWindowsTime</td>
<td>158</td>
</tr>
<tr>
<td>PointGetUnixTime</td>
<td>159</td>
</tr>
<tr>
<td>PointGetWindowsTime</td>
<td>160</td>
</tr>
<tr>
<td>PointMetadata</td>
<td>161</td>
</tr>
<tr>
<td>UnixLocalToUTC</td>
<td>163</td>
</tr>
<tr>
<td>UnixTimeToWindowsTime</td>
<td>164</td>
</tr>
<tr>
<td>UnixUTCToLocal</td>
<td>165</td>
</tr>
<tr>
<td>WindowsLocalToUTC</td>
<td>166</td>
</tr>
<tr>
<td>WindowsTimeToUnixTime</td>
<td>167</td>
</tr>
<tr>
<td>WindowsUTCToLocal</td>
<td>168</td>
</tr>
<tr>
<td>Regular Expression Methods from RegexSupport.g</td>
<td>169</td>
</tr>
<tr>
<td>Compile</td>
<td>170</td>
</tr>
<tr>
<td>CompileSubst</td>
<td>172</td>
</tr>
<tr>
<td>CompileSubstEx</td>
<td>173</td>
</tr>
<tr>
<td>Config</td>
<td>174</td>
</tr>
<tr>
<td>Exec</td>
<td>175</td>
</tr>
<tr>
<td>pcrs_job.Exec</td>
<td>177</td>
</tr>
<tr>
<td>GetStringNumber</td>
<td>178</td>
</tr>
<tr>
<td>Match</td>
<td>179</td>
</tr>
<tr>
<td>Study</td>
<td>181</td>
</tr>
<tr>
<td>Subst</td>
<td>182</td>
</tr>
<tr>
<td>Quality Name Function from Quality.g</td>
<td>183</td>
</tr>
<tr>
<td>GetQualityName</td>
<td>184</td>
</tr>
<tr>
<td>Classes from HistorianSupport.g</td>
<td>186</td>
</tr>
<tr>
<td>Historian</td>
<td>187</td>
</tr>
<tr>
<td>HistoryBuffer</td>
<td>192</td>
</tr>
<tr>
<td>HistoryValue</td>
<td>194</td>
</tr>
<tr>
<td>HistTest.g</td>
<td>195</td>
</tr>
<tr>
<td>Modbus Commands Methods from ModbusSupport.g</td>
<td>200</td>
</tr>
<tr>
<td>apply</td>
<td>201</td>
</tr>
</tbody>
</table>
addPoint .......................................................................................................................... 202
addRange .......................................................................................................................... 203
cancel .................................................................................................................................. 204
createSlave .......................................................................................................................... 205
deletePoint .......................................................................................................................... 206
deleteSlave .......................................................................................................................... 207
enableMaster ...................................................................................................................... 208
enableSlave .......................................................................................................................... 209
reloadSettings ..................................................................................................................... 210
slaveExists .......................................................................................................................... 211
A. Basic Troubleshooting ..................................................................................................... 212
B. License Copyright Information ........................................................................................ 213
Introduction

The Cogent DataHub has a powerful, built-in scripting language called Gamma. Using Gamma, you can interact with the DataHub and its data in various ways, such as:

- Run a script whenever a data point value changes.
- Build custom dashboards and summary displays,
- Create self contained DataHub applications.
- Create alarm condition scripts that display warning messages.
- Connect to ODBC compliant relational databases to extract data as well as create records from live data.
- Create server simulation programs to test production systems before you 'go live'.
What's Different About DataHub Scripting?

Scripts and their Environment

The DataHub scripting environment is different from most other programming paradigms because its primary purpose is to allow users to interact with the highly dynamic real-time process data that flows through the DataHub. Although syntactically similar to C, DataHub scripts offer unique capabilities, due in part to these features:

- **Dynamic Environment**
- **Event Driven**
- **Object Oriented**

Dynamic Environment

DataHub scripts run in a *dynamic environment*. The DataHub scripting engine, called Gamma, starts when the DataHub starts, and runs continuously until the DataHub shuts down. You can think of Gamma as a kind of processing power grid that's always switched on and running in the background. All of the data in the DataHub is available in this grid. DataHub users can tap into the power of the grid through scripts.

Scripts are like tools plugged into the grid. All of the live data in the DataHub is available to each script. A user script can be started manually or automatically, and typically runs until the DataHub shuts down. Scripts can range in complexity from the simple *Hello World* example in this manual to the entire Data Logging interface of the DataHub itself.

In addition to scripting, you can access Gamma interactively, from a command line in the **Script Log**.

Event Driven

DataHub scripts are *event-driven*, meaning that they respond to events as they occur. In our power grid analogy above, when you start a script, it's like switching on a tool that's
plugged into the grid. The tool sits in standby mode, ready to respond when needed. It has been programmed to respond in a particular way to certain events.

This behavior is difficult to achieve using a typical linear program that gets executed instruction by instruction. For example, a DataHub script has no mainline. Instead, a DataHub script contains two major elements:

1. **Event handlers** contain the code to be run when a DataHub point changes value.
2. The `.OnChange` method specifies the event (usually a data change) for which an event handler should be invoked. This change may be caused by an alarm condition or a timer firing or any other real-world event represented by a point in the DataHub. The `.OnChange` method binds a data change event in the DataHub to a specific event handler.

   Please refer to Partial Evaluation for an example.

When the script is run, Gamma loads all the bound event handlers into memory, and waits. Whenever a bound point in the DataHub changes value, Gamma executes the event handler code.

**Object Oriented**

DataHub user scripts are *object oriented*. Each user script creates a class derived from a base class called the `Application` class. This approach keeps the variables and methods of the script together in a tidy package, allowing them to be global in scope within the class, but separate from any other scripts running in Gamma. To make the scripts easy to write, the *New Script File* dialog from the *Scripting* option of the DataHub Properties window creates an instance of the `Application` class automatically, complete with templates for methods. Based on this template, a typical script contains the following:

1. A Gamma `require` function to load the `Application` class.
2. A class definition for the script's class. By default this class gets the same name as the script.
3. Method definitions for event handlers and other functions.
4. A constructor method, containing one or more calls to the `.OnChange` method of the `Application` class (see explanation above). This is as close to a "mainline" as a DataHub script gets, but in reality there is no linear flow to the program once all the code has been read in.
5. A destructor method, specifying any code that needs to be cleaned up when the class is destroyed or the script shuts down.
6. Class instantiation. Once the class has been instantiated, it just sits there and responds to events.

**Example**

Here is what a new, unedited template for the class `MyApp` would look like:
What's Different About DataHub Scripting?

/* All user scripts should derive from the base "Application" class */
require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or * Windows programming. Uncomment either or both. */
//require ("WindowsSupport");
//require ("ODBCSupport");

/* Applications share the execution thread and the global name * space, so we create a class that contains all of the functions * and variables for the application. This does two things: * 1) creates a private name space for the application, and * 2) allows you to re-load the application to create either * a new unique instance or multiple instances without * damaging an existing running instance. */
class MyApp Application
{
}

/* Use methods to create functions outside the 'main line'. */
method MyApp.samplemethod ()
{
}

/* Write the 'main line' of the program here. */
method MyApp.constructor ()
{
}

/* Any code to be run when the program gets shut down. */
method MyApp.destructor ()
{
}

/* Start the program by instantiating the class. If your * constructor code does not create a persistent reference to * the instance (self), then it will be destroyed by the * garbage collector soon after creation. If you do not want * this to happen, assign the instance to a global variable, or * create a static data member in your class to which you assign * 'self' during the construction process. ApplicationSingleton() * does this for you automatically. */
Symbols, Variables, and Evaluation

Complementing their dynamic environment and event-driven behavior, DataHub scripts have a slightly different approach to symbols and variables compared to other programming languages like C or Java.

Symbols and Variables

One of the fundamental units of Gamma, the DataHub scripting language, is a symbol. Symbols are made up of one or more alphanumeric characters, as well as "_". A symbol gets created whenever a unique group of characters appears in a script for the first time; from that point on the symbol is a unique object in Gamma. When first created, symbols are variables. The value of a variable can be assigned and reassigned at any point in the script. Variables that are not assigned a value have a default value of _undefined_.

Variable Scoping and Dynamic Typing

Similar to many programming languages, variables in Gamma can be local or global in scope. At the same time, to provide maximum flexibility, variables in Gamma are dynamically typed. Each time a variable is assigned a value, Gamma assigns or reassigns the type for that variable, based on the new value. This facilitates rapid development and eliminates the need to type or even declare all variables before they are used. Of course, good programming principles must still be observed when writing scripts to ensure that the variables are of the correct type for the circumstances.

The Read/Evaluate Cycle

When a DataHub script is run, Gamma first parses the script, reading and evaluating each statement in turn. For functions or methods, first each argument gets evaluated, and then each statement gets evaluated. Variables get evaluated to their value. Literals like numbers, strings, arrays, and so on get evaluated to themselves. This read/evaluation cycle iterates through the program on a recursive basis until the entire code gets read and evaluated. When the process is complete, Gamma executes the code.

Preventing Evaluation

Sometimes you might not want Gamma to evaluate a statement or variable when it parses the code. For example, when attaching an event handler to the .OnChange method, you don't want the code of the event handler to run until the event actually occurs. To prevent evaluation of a statement or variable, you can use the # character, a Gamma quote operator. Putting the # quote operator in front of any Gamma expression turns it into a literal, causing it to be evaluated to itself, as if it were a number or a string.

Example 1

This example shows an interactive session with Gamma in the Script Log. The ---> sym-
What’s Different About DataHub Scripting?

bol indicate where the user has input an expression, and the next line shows what Gamma has returned as the result of evaluation.

1. First we assign a value of 5 to the variable myvar:

   ```
   --> myvar = 5;
   5
   ```

   Gamma returns the value of myvar, which is 5.

2. Then we pass the variable myvar, to Gamma:

   ```
   --> myvar;
   5
   ```

   Gamma evaluates it and returns the value, 5.

3. Now we pass the variable myvar, to Gamma, this time quoted using the # symbol.

   ```
   --> #myvar;
   myvar
   ```

   And now Gamma evaluates myvar as its literal name, myvar. The expression itself has passed through the evaluator intact, without being evaluated.

**Partial Evaluation**

In some circumstances you might need Gamma to evaluate part of your statement, but not all of it. For this, there are two more quote operators. The `quote operator indicates that this statement should not be evaluated, except for those places marked by the @ quote operator.

**Example 2**

In this example, we use the Gamma list function to illustrate how partial evaluation works. The list function creates a space-separated list out of its arguments.

1. First, let’s define our variables and demonstrate the list function.

   ```
   --> myvar = 5;
   5
   --> yourvar = 9;
   9
   --> list(myvar, yourvar);
   (5 9)
   ```

   Gamma first evaluates the arguments of the list function to 5 and 9, then applies the list function and puts them into a list: (5 9).

2. Now let’s use the # quote operator:

   ```
   --> list(#myvar, #yourvar);
   (myvar yourvar)
   --> #list(myvar, yourvar);
   (list myvar yourvar)
   ```

   First we quoted the individual arguments, then the entire expression. Do you see the difference in the result? The second return value, (list myvar yourvar) illustrates
What's Different About DataHub Scripting?

the internal syntax of Gamma, which is Lisp. Lisp functions are always of this syntax:

\( \text{function_name \ arg_1 \ arg_2 \ldots} \).

3. Now, suppose we want to partially evaluate the expression. First, let's use the `quote operator alone:

\[ \text{---> `list(myvar, yourvar);} \]
\[ \text{(list myvar yourvar)} \]

This gives the same result as the `# operator, above. Now let's use the ` operator with the @ to allow partial evaluation, like this:

\[ \text{---> `list(@myvar, @yourvar);} \]
\[ \text{(list 5 9)} \]
\[ \text{---> `list(myvar, @yourvar);} \]
\[ \text{(list myvar 9)} \]

In the first line, we prevented the evaluation of the list function itself, but allowed Gamma to evaluate both of its arguments. In the second line, we allowed the evaluation of only one argument.

4. Here are a few more examples, incorporating the Gamma string function, which turns an expression into a string:

\[ \text{---> `string(list(myvar, yourvar));} \]
\[ "(5 9)" \]
\[ \text{---> `string(#list(myvar, yourvar));} \]
\[ "(list myvar yourvar)" \]
\[ \text{---> `string(`list(@myvar, @yourvar));} \]
\[ "(list 5 9)" \]
\[ \text{---> `string(list(@myvar, @yourvar));} \]
\[ (string (list 5 9)) \]
\[ \text{---> `string(@(list(myvar, yourvar)));} \]
\[ (string (5 9)) \]

In each case, the `quote operator prevents the evaluation of the overall expression, while the @ operator allows the evaluation of the sub-expression that immediately follows it.

5. How does this apply to events in a typical DataHub script? Here is an example, using the .OnChange method in a class named Example with a method called MethodA:

This the most important example, because this syntax is commonly used with the .OnChange method for handling events.

```plaintext
class Example Application
{
    ...
}

method Example.MethodA (x, y)
{
    ...
}
```
What's Different About DataHub Scripting?

In this example, we want to apply `MethodA` of our `Example` class to the values of variable `V1` and `V2` at the exact moment when `V1` changes its value. To do this we protect `V1` and `V2` from evaluation, using the `#` quote operator. We also do not want to evaluate the `MethodA` method, but we do have to evaluate the key variable indicating the class (`@self`). So we use the `@` operator to prevent the evaluation of the method, and the `#` operator to allow the key variable `self` to be evaluated. This way Gamma knows which class the `.MethodA` belongs to.

Forcing Evaluation

In some cases you might need to force Gamma to evaluate an expression. For this, you can use the Gamma `eval` function. For example:

```gamma
donextline
myvar = 5;
5
---> #myvar;
myvar
---> eval(#myvar);
5
```

Access to DataHub Points

The main purpose of writing DataHub scripts is to interact with the live data represented by DataHub points. Gamma, the DataHub scripting language, has a few special provisions for working with DataHub points. Understanding these will make your task much easier.

Point Names

First, a DataHub point name is not a legal symbol in Gamma. DataHub point names use a colon (`:`) to separate the point name from the domain name, and commonly use a dot (`.`) as well, like this: `DataSim:Sine` or `MyDomain:Plant1.Tank3.Valve2`. Colons and dots are not normally allowed in a Gamma variable name, so we use the Gamma `symbol character operator` `$` to turn the whole string of characters into a single, valid Gamma variable. For example, these expressions:

```gamma
donextline
$DataSim:Sine
```

are valid Gamma variables.

The `$` operator tells Gamma that all characters except white space (space, tab, newline, carriage return, form feed) and the the set: `[] () "`, `' are accepted in the symbol name.
To get any of the above characters in a symbol name, you need to precede that character by a backslash. So, for example, this DataHub point:

```
default:plant.water level[1]
```

would be written in a script like this:

```
$default:plant.water\ level\[1\]
```

In brief, you need to put a $ before any DataHub point name in a DataHub script, and use a backslash within the name before any whitespace or [](),";' character.

### Point Values

Often when we use a DataHub point as a variable in a script, we don’t want to evaluate it at the time that the code is being read. We want to simply quote it, and let it be evaluated when an event occurs. In these cases, we use the # quote operator when referring to a DataHub point, like this:

```
#$DataSim:Sine
```

Or, if the point name is within an expression that requires partial evaluation, the syntax would be like this:

```
@$DataSim:Sine
```

Please refer to the section called “The Read/Evaluate Cycle” for more details about evaluation, or to the section called “Accessing Data” for an example of this syntax in use.

### Point Timestamps and Qualities

In addition to the value of a DataHub point, you might need to know its timestamp or quality at the moment of an event. This information can be accessed through two special Gamma scripts: Time.g and Quality.g. You can require these scripts by adding a require function at the beginning of your script, like this:

```
require ("Application");
...
require ("Time");
require ("Quality");
...
```

The Time.g script offers a number of time-related functions, while the Quality.g file has a single function that converts a numerical quality code into a human-readable text string.

### ODBC and Windows Scripting

#### DataHub ODBC (Open Database Connectivity) Scripting

DataHub ODBC support allows the DataHub to interface with ODBC-compliant databases. Please refer to the DataHub ODBC Scripting manual for more information.
DataHub Windows Scripting

DataHub Windows scripting gives access to over 1700 of the most important classes used for programming in MS Windows, wrapped as Gamma classes. Please refer to the DataHub Windows Scripting manual for more information.
Getting Started

How to Run a Script

DataHub scripts run on Gamma, the DataHub scripting engine, which starts up whenever the DataHub starts and runs continuously as long as the DataHub runs. You can access DataHub scripts and scripting capabilities by pressing the **Scripting** button in the **Properties** window.

This will display the **Scripting and Customization** screen. The upper half of the screen shows the Gamma files currently configured in the DataHub:

To run an existing script for the first time, you will need to first add it to the list of scripts. To create a new script, please refer to the section called “Creating a Script”

1. To add a script to the list, press the **Add** button and choose the script from the file selector. Scripts are kept in a DataHub's `scripts` folder. Scripts that come with the DataHub are installed here (32-bit or 64-bit):

   ```
   C:\Program Files (x86)\Cogent\Cogent DataHub\scripts\ 
   C:\Program Files\Cogent\Cogent DataHub\scripts\ 
   C:\Users\Username\AppData\Roaming\Cogent DataHub\scripts
   ```

   All content in this directory will be replaced by the default content when the DataHub is re-installed. If you plan to edit one of these scripts, or to write a new script, you should keep it in this folder for user-created scripts:

   ```
   C:\Users\Username\AppData\Roaming\Cogent DataHub\scripts
   ```

2. If you need to edit the script before running it, press the **Edit** button to open the selected script in the **Script Editor**.

3. To run the script manually, press the **Load Now** button.

4. To see any script output or error messages, you can press the **Script Log** button near the bottom of the Properties window to open the **Script Log**.
5. To configure a script to run automatically at startup, check the checkbox next to it. The next time you start the DataHub, this script will load and run automatically.

6. Once a script is started, it will continue running until the DataHub shuts down. To stop the script without shutting down the DataHub, press the **Script Manager** button to open the **Script Application Manager**.

![Script Application Manager](image)

Highlight the script you want to stop, and press the **Stop Selected** button.

**The Script Editor**

The DataHub comes with a built-in Script Editor\(^1\). You can open the Script Editor from the **Scripting** option of the Properties window of the DataHub. Select a filename, press the **Edit** button, and the Script Editor will open:

![Script Editor](image)

The Script Editor offers basic script editing features such as context-sensitive highlighting, prompted fill-ins for functions and variable names, automatic indenting, text string

---

\(^1\)This editor is based on the Scintilla and SciTE editor.
searches, and so on. In addition to the normal menu and toolbar options, it has a **Script** menu that provides the following options:

- **Evaluate Section** sends whatever block of text you have selected to the Gamma interpreter for immediate processing.

- **Reload Whole File** sends the entire file to the Gamma interpreter for immediate processing, without you having to save the file first. This functionality is also activated by pressing the blue arrow icon on the toolbar.

- **Check Syntax** checks the syntax of the whole file without running the script. Any errors will be displayed in the **Script Log**.

- **Open Script Log** opens the **Script Log**.

The Script Editor toolbar has a blue arrow icon in addition to three standard icons for creating, opening, and saving files. Clicking the blue arrow saves the script as written and runs it.

### The Script Log

The Script Log displays output from Gamma scripts, and can also be used to conduct interactive sessions, like a terminal. You can open this window in either of two ways:

- Click the **Script Log** button in the Properties window.

The Script Log should appear on your desktop:
You can use the text entry field at the bottom to send code to Gamma. Try the following:

1. Type: `a = 5;` and press **Enter**
   You should see the following on the screen:
   ```
   --> a = 5;
   5
   ```
   You have just created a *symbol* (`a`) and assigned it a value (5). That symbol is a variable, and is now available to Gamma until the DataHub shuts down. Notice that the Script Log inserts a prompt (`-->`) and shows your command to help you identify what you typed in.

2. Press the **Clear** button to clear the Script Log. Press the **Close** button to close the Script Log window, then reopen it.

3. Type: `a;` and press **Enter**
   You should see the following on the screen:
   ```
   --> a;
   5
   ```
   Sure enough, the value of `a` is still in Gamma

4. Type: `princ("Hello world.\n")`; and view the results:
   ```
   --> princ("Hello world.\n");
   Hello world.
   t
   ```
   Why the `t`? It is the return value from the `princ` function, a *logically true value*. Every Gamma function returns a value. The string 'Hello world.' is the byproduct or result of running the function, but the actual return value is `t`. For more details on Gamma programming, please refer to the Gamma manual.

5. Now, let's see a value in the DataHub. Start DataSim, then type: `$DataSim:Sine;` and press **Enter**. You should see something like this:
   ```
   --> $DataSim:Sine;
   ```
-0.47552825816976968

This was the value of the Sine point in the DataSim domain of the DataHub at the moment you pressed the Enter key.

The colon character (:) is used to divide the domain name from the point name. The dollar sign character ($) tells Gamma that the colon is part of the variable name, not a syntactic element.

6. You can re-enter up to the last 10 commands by pressing the down arrow on your keyboard. Try it now. Press the down arrow until you see the last command, $DataSim:Sine;, and press Enter. Try it several times. You will get different values because the DataSim program is running.

This gives you a taste of working with Gamma, but to accomplish anything really useful and to save your work, you'll need a script. The following sections will explain how to access and edit scripts, and create your own.

The Script Application Manager

The Script Application Manager lets you view the scripts currently running in the DataHub, and stop selected scripts if desired. You can open the Script Application Manager by pressing the Script Manager button from the Scripting option of the DataHub Properties window.

To stop a script, highlight it, and press the Stop Selected button.

The columns display the following information:

- **Class**: the name of the instance of the Application class created in the script.
- **Timers**: the number of timers active in this script.
- **Change Events**: the number of change events active in this script.
- **Menu Bindings**: the number of menu entries that this script has placed in the system tray menu.
- **Sub-Menu Name**: the name of the submenu in the system tray menu into which this script has placed its menu entries.
Writing Scripts

Writing a script for the DataHub is not difficult, particularly when you follow a few basic principles. Working directly with DataHub points and the Gamma interpreter environment creates special opportunities, and you can take best advantage of them by using the suggestions offered here.

Most of the examples in this chapter use DataSim, which is installed with the DataHub. You should ensure that it is connected and sending data to the DataHub before attempting these examples. For more information, please refer to DataSim in the Cogent DataHub manual.

Creating a Script

The best way to write a DataHub script is to create a single class in which your entire script runs. This keeps all variables and functions (methods) local to the class and isolated from any other script running in Gamma. Rather than create the class from scratch, the DataHub writes a template for you that contains what you need. Here's how it works:

1. Open the DataHub Properties window and select the Scripting option.
2. Click the New button. This dialog will appear:

   ![New Script File dialog box]

3. In the New class: field, enter a name of your choice. The default is MyApp.
4. Look at the File name: field to make sure this is where you want the script to be created. If not, you can browse your file system for a better location.
5. Checking the Allow only one instance box will ensure that each instance of the class gets destroyed whenever a new instance is created. Thus you will only ever have one instance. If you want your code to create multiple simultaneous instances of the class, don't check this box.
6. The Include sample code option lets you choose one of the following:
   - None will generate no sample code.
• **Windows** will generate code to help you create windows.

• **Data Manipulation** will generate code for doing linear transformations and other data manipulation.

These options will be illustrated and discussed in upcoming sections.

7. Click the **OK** button to create the file.

The **Script Editor** should open, displaying a script template ready for editing. The basic template looks like this:

```plaintext
/* All user scripts should derive from the base "Application" class */

require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or Windows programming. Uncomment either or both. */

//require ("WindowsSupport");
//require ("ODBCSupport");

/* Applications share the execution thread and the global name space, so we create a class that contains all of the functions and variables for the application. This does two things:
  1) creates a private name space for the application, and
  2) allows you to re-load the application to create either a new unique instance or multiple instances without damaging an existing running instance. */

class MyApp Application
{
}
```
Writing Scripts

/* Use methods to create functions outside the 'main line'. */
method MyApp.samplemethod ()
{
}

/* Write the 'main line' of the program here. */
method MyApp.constructor ()
{
}

/* Any code to be run when the program gets shut down. */
method MyApp.destructor ()
{
}

/* Start the program by instantiating the class. If your constructor code does not create a persistent reference to the instance (self), then it will be destroyed by the garbage collector soon after creation. If you do not want this to happen, assign the instance to a global variable, or create a static data member in your class to which you assign 'self' during the construction process. ApplicationSingleton() does this for you automatically. */
ApplicationSingleton (MyApp);

The following sections explain how to edit this template and run the finished script.

Hello World

This simple example demonstrates how to edit a new script by editing the .constructor method.

Edit the Constructor

1. Create a new script whose main class is called 'HelloWorld'. Here's how.
2. The body of a typical script is written as part of the .constructor method. Find the following lines to start editing:
   /* Write the 'main line' of the program here. */
   method HelloWorld.constructor ()
   {
   }

3. In between the brackets of the .constructor method, enter the following:
   pri
Notice that a drop-down box appears:
This box will appear any time you begin to write a function or variable name that is already available in Gamma.

4. Continue writing:

```javascript
princ()
```

The box now shows you the `princ` function syntax, in this case a symbolic expression (`s_exp`), commonly known as a symbol. Please refer to the Gamma manual for more information about function syntax and arguments.

5. Continue editing the `princ` function until your `.constructor` function looks like this:

```javascript
/* Write the 'main line' of the program here. */
method HelloWorld.constructor ()
{
    princ("Hello world.\n");
}
```

Run the Script

Click the blue arrow icon in the Script Editor toolbar to run the script, and then check the results in the Script Log window:

If you don’t get a ‘Hello world’ string in the Script Log, see Appendix A, Basic Troubleshooting.

Evaluate a Selection

Here’s a way to evaluate just a part of your code:

1. In the Script Editor, use the cursor to select just the text:

   ```javascript
   princ("Hello world.\n");
   ```

2. From the Script menu, select Evaluate Selection.
   You should see the string ‘Hello world.’ appear in the Script Log.

This feature of the Script Editor lets you run any part of a script without running the whole thing.
Accessing Data

A DataHub script has complete access to all the data in the DataHub. Every point in the DataHub is available in Gamma as a global variable, with the syntax:

```
$domain_name:point_name
```

The dollar sign character ($) tells Gamma that the colon is part of the variable name. A colon in Gamma is normally a syntactic element and can't be used in a variable name. However, the DataHub uses a colon (:) in point names to separate the domain name from the rest of the point name. So we need to use the dollar sign. The dollar sign is not part of the variable name.

For these examples, ensure that the DataSim program is running and connected to the DataHub.

Access a Value Once

To start with, here's how to read a value one time.

1. Create a new script with a class named AccessData, selecting Allow only one instance. Here's how.
2. Scroll down to the .constructor method, and enter the following:
   ```
   method AccessData.constructor () {
     princ("The sine is: ", $DataSim:Sine, "\n");
   }
   ```
3. Make sure the DataSim program is running, and that the Script Log is open so you'll be able to see the script output.
4. Click the blue arrow icon in the Script Editor toolbar. You should see a value appear in the Script Log window, like this:
```
   The sine is: -0.4755086581697696
```
5. Click the blue arrow icon several more times. Each time, a new value should appear in the Script Log. The values will differ because the value for the sine wave in DataSim is constantly changing.

Access Values Continuously

Getting a single value from the DataHub is fine, but we can do much better. Let's print each new value from a point every time it changes.
1. First you should move the `princ` statement out of the `constructor`. This may seem trivial in this example, but it is a good habit to get into. Move the `princ` statement to the `AccessData.sample` method, and edit the method name and `princ` statement to look like this:

```plaintext
method AccessData.print_point (pt)
{
    princ("The sine is: ", pt, ",\n";)
}
```

2. Edit the `constructor` method like this:

```plaintext
method AccessData.constructor ()
{
    .OnChange(#$DataSim:Sine,
    `(@self).print_point($DataSim:Sine));
}
```

The `OnChange` method is inherited from the parent class, `Application`. This method is a wrapper for the `on_change` function, which tells Gamma to evaluate an expression when a given symbol changes value. In this case, the symbol is `$DataSim:Sine` and the expression is our `print_point` function. The first expression is protected from evaluation by the `#` quote operator.

The second expression is also quoted so that it doesn't run until the point actually changes. However, the `self` that is usually understood must be explicitly written and evaluated so that Gamma knows what the `.print_point` method applies to. Therefore, we use the `\` and `@` quote operators.

3. Click the blue arrow icon in the Script Editor toolbar.

A new value gets written twice a second. To see it really fly, bring up the DataSim window, click the `More` button, and change the `Update Frequency` to 200 (don't forget to click `Apply Changes`). To make it stop, click DataSim's `Pause` button.

But what if you can't stop the data flow? How do you get the `princ` function to stop?

### Stop Accessing Values

Often you will want a script's change functions to stop when the class instance is destroyed. Conveniently, the `Application` class has a destructor that runs any time a child class gets destroyed, and which removes all `OnChange` functions. One way to destroy the instance of class is with a timer.

```plaintext
after(3, `destroy(@self));
```

This uses Gamma's `after` timer function, which in this case will activate 3 seconds after the instance is constructed and cause the instance to destroy itself. Again, to pre-
vent the `destroy` function from being evaluated and destroying our instance prematurely, we have to quote it. And again, we use the `:` and `@` quote operators to evaluate the `self` argument. After 3 seconds, the `destroy` function will be evaluated, and the instance gets destroyed.

2. Your two methods should now look like this:

```plaintext
method AccessData.print_point (pt)
{
    princ("The sine is: ", pt, "\n");
}

method AccessData.constructor ()
{
    .OnChange(#$DataSim:Sine, `(@self).print_point($DataSim:Sine));
    after(3, `destroy(@self));
}
```

3. Run the script. You should see output in the Script Log for 3 seconds, then nothing.

**Verify Results**

1. To verify that the instance of the class has been destroyed, type `instance_p(_AccessData_Singleton)` in the text entry field at the bottom of the Script Log and press `Enter`. The output should look like this:

   ```plaintext
   --> instance_p(_AccessData_Singleton);
   nil
   ```

   The `instance_p` function is a Gamma predicate that checks to see if `accessdata` is an instance. Gamma returns `nil`, indicating that it is not an instance.

2. Since `accessdata` was an instance and is no more, it is probably a destroyed instance. To check, type `destroyed_p(_AccessData_Singleton);` and press `Enter`. The output should look like this:

   ```plaintext
   --> destroyed_p(_AccessData_Singleton);
   t
   ```

   Gamma returns `t`, indicating that it is a destroyed instance.

**Modifying Data**

DataHub scripting lets you modify data as it passes through the DataHub. This example script makes a linear transformation, writes the results to a different DataHub point, and incidentally prints the value of `point` and the conversion.

1. Create a new script whose main class is called `ModifyData`. Here's how.

2. Change the name of `.samplemethod` to `.convert` and edit it like this:

   ```plaintext
   method ModifyData.convert (pt1, !pt2, multiplier, adder)
   {
       local output;
   }
   ```
Writing Scripts

set(pt2, ((pt1 * multiplier) + adder));
princ(format("The sine is: %2.3f  Converted it is: %2.3f\n", pt1, eval(pt2)));
}

This method does the conversion and prints the output. Notice that the pt2 argu-
ment has an exclamation point (!) in front of it. This protects the argument from be-
ing evaluated, because we only want the point name, not its value. Please refer to
Function Arguments in the Gamma manual for more information.

We need to assign a value to the point, but keep the variable name associated with
the DataHub point, not the value. This requires the set function, because using the
equals sign would just assign the value to the variable, and it would never reach the
DataHub point.

3. Edit the .constructor method to look like this:

method ModifyData.constructor ()
{
    multiplier = 3;
    adder = 5;

    if (undefined_p($default:ConvertedSine))
        datahub_command("(cset default:ConvertedSine "\")", 1);

    .OnChange(#$DataSim:Sine, 
        `((@self).convert($DataSim:Sine, 
            $default:ConvertedSine, 
            multiplier, adder));

    after(3, `destroy(@self));
}

We don't make the multiplier and adder variables local because they'll be used by
the Application class's .destructor method. We use the datahub_command
function call the cset function to have the DataHub create the ConvertedSine point
and set its value to an empty string. We then use the inherited .OnChange method to
set up the .convert method.

4. The script is ready to run. Open the Data Browser window right-clicking the DataHub
icon in the System Tray and selecting View Data from the pop-up menu. Select the
default domain. Also make sure the DataSim program is running and the Script Log is
open.

5. Run the script. It should run for 3 seconds, writing data to the Script Log, and chang-
ing the values to the point ConvertedSine in the default domain.

Of course, the data printed in the Script Log is just for convenience. You could com-
ment out the princ statement, or remove it altogether. The main thing is the data
updating in the Data Browser.

Making a Window
DataHub scripting offers many of the Windows classes wrapped as Gamma classes. Thus you can create windows, buttons, entry forms, tabs, dialogs, and so on. The DataHub Windows Scripting manual contains more information. This is how you create a basic window.

1. Open the Properties window, select the **Scripting** option, and click the **New** button to create a new script.
2. In the New Script File dialog, name the main class 'MyWindows' and select the **Windows** option. More details.
3. Add the file to your list of files, and load it now. Here's how. A new window will open:

![Image of a basic window](image)

4. Click the close icon in the top right corner. The window will close, and you will see this dialog box:

![Image of a dialog box](image)

This is an example of a basic window. Here are the main parts of your `MyWindows.g` script:

```g
class MyWindows Application
{
    window;
}
```

The window itself is an instance variable of the `MyWindows` class.
{ 
  local   rect = CreateRect (0, 0, 300, 300), txt;  
  .window = new GWindow();  
  
  .window.Create (0, rect, "Hello", WS_OVERLAPPEDWINDOW, 0);  
  .window.CenterWindow();  
  txt = .window.CreateControl (GStatic, 0, 0, 280, 22,  
                             "Hello world", SS_CENTER);  
  txt.CenterWindow();  
  .window.MessageHandler (WM_DESTROY, `destroy(@self));  
  .window.ShowWindow (SW_SHOW);  
}

method MyWindows.destructor ()
{
  // The WM_DESTROY message could come before or after this  
  // destructor depending on whether the application instance  
  // is destroyed or the window is closed first. We protect  
  // against the case where the window is closed first.  
  if (instance_p(.window) && .window.GetHwnd() != 0)  
    .window.PostMessage (WM_CLOSE, 0, 0);  
  MessageBox(0, string ("Application: ", class_name(self),  
                      " completed."), "Done", 0);  
}

More information about Windows scripting and the Windows classes and methods can be found in the [DataHub Windows Scripting](#) manual.

**Encrypting a Script**

The DataHub archive contains a script called `EncryptScript.g` that makes an encrypted copy of any Gamma script. You can encrypt your scripts as follows:

1. Run the `EncryptScript.g` script. [Here's how.](#)
2. Right-click on the DataHub icon in the system tray to open the DataHub menu. You should see a new menu entry: [Script Encryption](#).
There are two options for encrypting a file. The Encrypt File option uses 8-bit characters, which produces a smaller, faster loading file than an ASCII file, but it may not transfer properly through some mail or FTP servers. If that is an issue, you can use Encrypt File (ASCII).

3. Choose either Encrypt File or Encrypt File (ASCII). This will open a file selection dialog.

4. Select the Gamma (.g) file that you want to encrypt. If the encryption succeeds, you will get a success message. The encrypted script will have the same name as the original script, except with a .gmc extension.

5. You can add the encrypted script to your list of scripts now, and run it. Here’s how. The encrypted script should behave exactly the same as the .g file, but will be difficult for a user to examine.

Scripting Tips

Here are some ways to facilitate scripting.

Copying a complete tutorial

The code from DataHub tutorials can be copied directly from the text into the Script Editor. Here’s how to make a new script from a complete tutorial.

1. Open the DataHub Properties window and select the Scripting option.
2. Click the New button. This dialog will appear:
3. In the **New class**: field, enter a name of your choice.
4. Click the **OK** button to create the file. The **Script Editor** will open, displaying a script template.
5. Using the cursor or **Ctrl-A**, select all the text and delete it.
6. Copy a complete tutorial from the DataHub Windows Help manual or online documentation into the Script Editor.
7. Save and run the script using the blue arrow icon ▶️.

**Setting up a scripting environment**

If you are working on one script regularly and need to restart the DataHub often, here's a way to automatically open the Properties window, the Script Log and the Script Editor with your file loaded.

1. Open the Properties window and Script Log.
2. Position them on the screen, then close them. They will come up in the same place the next time you open them. (The Script Editor does not yet have this feature.)
3. Write a script of these three lines:
   ```lisp
   datahub_command ("(show_properties 1)");
datahub_command ("(show_script_log 1)");
edit_file ("c:\projects\myfile.g");
   ```
   Two slash marks (\) are necessary—the first slash mark escapes the second one.
4. Add this script to the list of scripts in the DataHub, and check the activation box beside it. Then next time you start the DataHub, the properties and script log windows will come up where they were before, and an editor will be opened on the file `c:\projects\myfile.g`.
5. When you don't want the script to auto-load, just un-check the activation box.
The Application class

The Application class is the parent class for all the applications you create with the New button in the Scripting option of the Properties window. It provides an environment that makes it easy for you to program for changes and events. It also allows you to set up timers, and to add menus for your scripts to the DataHub's pop-up system tray menu. This chapter contains an overview of the Application.g file, where the Application class and its methods are defined.

Class Definition

This code defines the base Application class:

```c
/* A base application class that keeps track of change functions 
and removes them when the object is destroyed */

if (undefined_p(Application) || !class_p(Application))
{
    class Application
    {
        _ChangeFunctions;
        _TimerIDs;
        _MenuActions;
        _SubMenus;
        static:
        _Instances;
        _MenuItemID = 20000;
        _MenuItems;
        _TraySubmenu;
        _AllMenuActions;
        _TrayMenuPosition = 6;
    }
}
```

Before defining the class, we test to see if it already exists, using the Gamma predicates undefined_p and class_p functions in an if statement. We want only a single instance of the class so that there is only one copy of the _Instances static variable in the system. This _Instances variable tracks the currently loaded applications, and we do not want several different copies of it floating about.

The instance variable _ChangeFunctions is a list of all the change functions that are defined by the class's .OnChange method. The _TimerIDs, _MenuActions, and _Submenus instance variables are similar lists for any timer IDs, menu actions, and custom submenus that may be defined for the class.
Construction and Destruction

The constructor and destructor methods help with general class housekeeping.

```java
method Application.constructor ()
{
    ._Instances = cons (self, ._Instances);
}
```

The constructor adds each instance of the class to the list of all the class instances currently defined in Gamma. The Gamma `cons` function is used to build the list.

```java
method Application.destructor ()
{
    local id;

    ._Instances = remove (self, ._Instances);
    .RemoveAllEventHandlers ();
    .RemoveAllMenus ();

    if (!._AllMenuActions)
    {
        .RemoveSystemMenu();
    }
}
```

When an instance of the class gets destroyed, this destructor method will be run first. It removes this instance of the class from the `_Instances` list using the Gamma `remove` function. Then it removes all event handlers and menus, using the `RemoveAllEventHandlers`, `RemoveAllMenus`, and `RemoveSystemMenu`, as applicable.

Handling Events

Event handlers are bound to events using the `OnChange` method.

```java
method Application.OnChange (sym, fn)
{
    local chfn = cons (sym, fn);
    ._ChangeFunctions = cons (chfn, ._ChangeFunctions);
    on_change (sym, fn);
    chfn;
}
```

The `OnChange` method is a wrapper for the `on_change` function that does the actual binding of the event handler. First the `OnChange` method creates a two-member list `(chfn)` consisting of a symbol (sym) and the function that should run (fn) when the symbol changes value. That short list is added to the class's `_ChangeFunctions` list. All lists are
constructed using the Gamma `cons` function. Finally, the `on_change` function links the symbol to the event-handling function. What gets returned, `chfn`, is a two member list—exactly what the unwrapped `on_change` function would have returned.

One way to remove an event handler is with the `RemoveChange` method.

```
method Application.RemoveChange (chfn)
{
    ._ChangeFunctions = remove (chfn, ._ChangeFunctions);
    remove_change (car(chfn), cdr(chfn));
}
```

This is a wrapper on `remove_change`, to be used when you need to remove just a single function from `._ChangeFunctions` rather than all of them. See also `RemoveAllChanges` and `RemoveAllEventHandlers`.

**Timers**

There are three different methods for attaching a timer to an event-handling method or function: `TimerAt`, `TimerAfter`, and `TimerEvery`. They use the Gamma functions `at`, `after`, and `every` respectively. For example:

```
method Application.TimerEvery (seconds, fn)
{
    local tid = every (seconds, fn);
    ._TimerIDs = cons (tid, ._TimerIDs);
    tid;
}
```

This is a wrapper on the Gamma `every` function. It creates a timer ID and adds it to the class's list of timer IDs (`._TimerIDs`). This way the timer can be identified and removed when necessary.

```
method Application.RemoveTimer(tid)
{
    if (find_equal (tid, ._TimerIDs))
    {
        cancel (tid);
        .droptimer (tid);
    }
}
```

This method uses the Gamma `find_equal` function to locate a timer according to its ID number (`tid`), and then uses the Gamma `cancel` function to cancel it.

Timers created using `TimerAt`, `TimerAfter`, and `TimerEvery` are automatically cancelled when the `Application` instance is destroyed.
Menus

There are a number of methods in the Application class that you can use to create sub-menus and menu items on the DataHub pop-up system tray menu. Rather than examining the code, let us look at what kinds of menus can be generated. Each of the menus below has two regular items that simply print their name in the Script Log, as well as an "Exit" item that shuts down the script. A complete, working script containing all of these examples is given below. All of the methods used here are documented in Methods and Functions from Application.g.

Putting Menu Items in the Scripts Submenu

The methods used here cause a Scripts submenu to be created on the main DataHub pop-up menu, and attach items directly to that submenu. This code:

```
.AddCustomMenuItem("Base 1", princ("Base 1\n"));
.AddCustomMenuItem("Base 2", princ("Base 2\n"));
.AddStopMenuItem("Base Exit");
```

Would add these menu items.

Creating a Submenu using Convenience Methods

The methods used here are convenience methods that wrap the standard methods used below. They create a submenu on the main DataHub menu, attaching menu items to the immediately preceding parent, in the order written in the code. This code:

```
.AddCustomSubMenu("Custom SubMenu", 3);
.AddCustomMenuItem("Custom Item 1-1",
   princ("Custom Item 1-1\n"));
.AddCustomMenuItem("Custom Item 1-2",
   princ("Custom Item 1-2\n"));
.AddStopMenuItem("Custom 1 Exit");
```

Would put up this submenu.
Creating Submenus on the Main Menu

The methods used here are the standard methods used to create submenus on the main DataHub menu. Each menu item is identified by its parent, so menu items can be added in any order. This code:

```c
local mymenu, mymenu2;
mymenu = .AddSubMenu(get_tray_menu(), 3, "Normal SubMenu");
mymenu2 = .AddSubMenu(get_tray_menu(), 4, "Normal SubMenu 2");
.AddMenuItem(mymenu, -1, "Menu Item 1",
    `princ("Menu Item 1-1
"));
.AddMenuItem(mymenu2, -1, "Menu Item 1",
    `princ("Menu Item 2-1
"));
.AddMenuItem(mymenu, -1, "Menu Item 2",
    `princ("Menu Item 1-2"
"));
.AddMenuItem(mymenu2, -1, "Menu Item 2",
    `princ("Menu Item 2-2"
"));
.AddMenuItem (mymenu, -1, "Normal 1 Exit", `destroy (@self));
.AddMenuItem (mymenu2, -1, "Normal 2 Exit", `destroy (@self));
```

Would put up two submenus, the first of which is open here.

The `menutest.g` script

This script contains all of the above examples.

```c
/* All user scripts should derive from the base "Application" class */
require ("Application");
```
/* Get the Gamma library functions and methods for ODBC and/or Windows programming. Uncomment either or both. */

//require ("WindowsSupport");
//require ("ODBCSupport");

/* Applications share the execution thread and the global name space, so we create a class that contains all of the functions and variables for the application. This does two things:
   1) creates a private name space for the application, and
   2) allows you to re-load the application to create either a new unique instance or multiple instances without damaging an existing running instance.
*/
class menutest Application
{
}

/* Use methods to create functions outside the 'main line'. */

/* Create submenus and add items to them, using the AddCustom* convenience methods. Note that the items are added in sequential
order for each menu. */
method menutest.custom()
{
   .AddCustomSubMenu("Custom SubMenu", 3);
   .AddCustomMenuItem("Custom Item 1-1",
      `princ("Custom Item 1-1\n")
   .AddCustomMenuItem("Custom Item 1-2",
      `princ("Custom Item 1-2\n")
   .AddStopMenuItem("Custom 1 Exit");

   .AddCustomSubMenu("Custom SubMenu 2", 4);
   .AddCustomMenuItem("Custom Item 2-1",
      `princ("Custom Item 2-1\n")
   .AddCustomMenuItem("Custom Item 2-2",
      `princ("Custom Item 2-2\n")
   .AddStopMenuItem("Custom 2 Exit");
}

/* Create regular submenus and add items to them, using the Add* methods. Note that the items can be added in any order. */
method menutest.direct()
{
   local    mymenu, mymenu2;
   mymenu = .AddSubMenu(get_tray_menu(), 3, "Normal SubMenu");
mymenu2 = .AddSubMenu(get_tray_menu(), 4, "Normal SubMenu 2");
.AddMenuItem(mymenu, -1, "Menu Item 1",
`princ("Menu Item 1-1\n"));
.AddMenuItem(mymenu2, -1, "Menu Item 1",
`princ("Menu Item 2-1\n"));
.AddMenuItem(mymenu, -1, "Menu Item 2",
`princ("Menu Item 1-2\n"));
.AddMenuItem(mymenu2, -1, "Menu Item 2",
`princ("Menu Item 2-2\n"));
.AddMenuItem (mymenu, -1, "Normal 1 Exit",
`destroy (@self));
.AddMenuItem (mymenu2, -1, "Normal 2 Exit",
`destroy (@self));
}

/* Write the 'main line' of the program here. */
method menutest.constructor ()
{
    /* Add menu items to a "Scripts" submenu that gets created automatically. */
    .AddCustomMenuItem("Base 1", `princ("Base 1\n");
    .AddCustomMenuItem("Base 2", `princ("Base 2\n");
    .AddStopMenuItem("Base Exit");

    /* Create the normal menus. */
    .direct();

    /* Create the custom menus. */
    .custom();
}

/* Any code to be run when the program gets shut down. */
method menutest.destructor ()
{
}

/* Start the program by instantiating the class. If your constructor code does not create a persistent reference to the instance (self), then it will be destroyed by the garbage collector soon after creation. If you do not want this to happen, assign the instance to a global variable, or create a static data member in your class to which you assign 'self' during the construction process. ApplicationSingleton() does this for you automatically. */
ApplicationSingleton (menutest);
Example Scripts
LogFile.g

LogFile.g — logs data to a text file when a point changes value.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called "How to Run a Script" for more information on using scripts.

/* This script shows how to log data to a text file. It uses a trigger point to signal an alarm condition (non-zero), which causes a value to be written to the file. 
* To use this script with your points, replace 'default:triggerpt' in the LogFile class with your trigger point, and replace 'default:loggedpt' with the point whose value you wish to log. 
* You can also change the name of the log file.
*/

/* All user scripts should derive from the base "Application" class */

require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or Windows programming. Uncomment either or both. */

require ("WindowsSupport");
//require ("ODBCSupport");

/* Applications share the execution thread and the global name space, so we create a class that contains all of the functions and variables for the application. This does two things: 1) creates a private name space for the application, and 2) allows you to re-load the application to create either a new unique instance or multiple instances without damaging an existing running instance. */

class LogFile Application {
// The trigger point whose value determines when logging takes place.
trigger = #$default:triggerpt;

// The point whose value gets logged.
logged = #$default:loggedpt;

// The name of the log file.
log_file_name = "c:/tmp/logfile.txt";

// The file handle to the open file
log_file;
}

/*
 * This method writes the trigger value and the logged point value
 * for alarm conditions and non-alarm conditions. The first argument
 * is the actual value of the trigger point, and the second is the
 * symbolic name of the point to be logged.
 */
method LogFile.AlarmOccurred(triggervalue, !logpoint)
{
    local value = eval (logpoint);
    if (triggervalue != 0)
    {
        writec (.log_file, format ("Alarm: %-20s = %10g\n", string(logpoint), value));
        princ ("Alarm condition: ", logpoint, ", ", value, ");
    }
    else
    {
        writec (.log_file, format ("Cleared: %-20s = %10g\n", string(logpoint), value));
        princ ("No alarm: ", logpoint, ", ", value, ");
    }
    flush (.log_file);
}

/* Write the 'main line' of the program here. */
method LogFile.constructor ()
{
    /* If the trigger and logged points don't exist in the DataHub,
     * create them. */
    datahub_command (string ("(create ", .trigger, " 1)"), 1);
    datahub_command (string ("(create ", .logged, " 1)"), 1);
/* Attempt to open the log file. */
.log_file = open (.log_file_name, "a");
if (!.log_file)
{
    MessageBox (0, string ("Could not open alarm log file: ",
        .log_file_name),
        "Error opening file", 0);
}
else
{
    /* Log the data whenever the trigger point changes. */
    .OnChange (.trigger, `(@(self).AlarmOccurred (value,
        @.logged));
}

/* Any code to be run when the program gets shut down. */
method LogFile.destructor ()
{
    if (.log_file)
        close (.log_file);
}

/* Start the program by instantiating the class. If your
* constructor code does not create a persistent reference to
* the instance (self), then it will be destroyed by the
* garbage collector soon after creation. If you do not want
* this to happen, assign the instance to a global variable, or
* create a static data member in your class to which you assign
* 'self' during the construction process. ApplicationSingleton()
* does this for you automatically. */
ApplicationSingleton (LogFile);
ReadCSV.g

ReadCSV.g — reads a CSV file and writes the points and values to the DataHub.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called "How to Run a Script" for more information on using scripts.

/*
 * This script reads a CSV file and writes the values found there
 * into a set of data points in the DataHub. The format of the
 * file is:
 * *
 * row 1:  name1, name2, name3, ...
 * row 2:  value1, value2, value3, ...
 * row 3:  value1, value2, value3, ...
 * ...
 * row N:  value1, value2, value3, ...
 * *
 * The script will read all rows in the file, but ignore all but
 * the first and last. The first row contains the point names and
 * the last contains the most recent data.
 * If a name is left blank then that column is ignored.
 * If a point name does not contain a domain name, then the domain
 * set in the "domain" member of the application is used.
 * *
 * e.g.,
 * default:point1, default:point2, default:point3
 * 1, 2, 3
 * 4, 5, 6
 *
 * will result in:
 * default:point1 = 4
 * default:point2 = 5
 * default:point3 = 6
 *
 * Strings containing ',' characters must be quoted within double
 * quotes, like this:
 * "hello, friend"
 */
* Double-quotes within strings must be escaped, like this:
  * "He said, \"hello\"."
  *
* This script will guess whether a value is a number or a string.
* If the value can be parsed to a number, it is treated as a number.
* Otherwise it is a string.
 *
* This script looks for new data at a set time interval.
 *
* This script will operate in one of two modes:
* In "reload" mode, the file is re-read from the beginning on each timer tick.
* In "append" mode, the file is kept open, and the file is read from the last read position on each timer tick. This mode will not work if the writing application does not open the file as "shared".
 *
* This script adds a menu item to the OPC DataHub system tray icon that allows the user to re-load the file, change reade mode, and toggle logging to the Script Log window.
 */

```
require ("Application");

class ReadCSV Application
{
    mode = #reload;  // set to #reload or #append
    domain = "default";
    filename = "c:/tmp/data.csv";
    verbose = t;
    update_secs = 5;
    separators = ","; // e.g, use "," for space separated, // or "\t" for tab-separated

    /* --- No need to change these --- */
    columns;
    fptr;
    modemenu;
    verbosemenu;
}

/* Logging function that prepends the time to the output. */
method ReadCSV.Log (args...)
{
    if (.verbose)
    {
        
```
funcall (princ, cons (date(), cons(" ": ", args)));
princ("\n");
}
}

/* Open the given file, if possible. */
method ReadCSV.OpenFile (filename)
{
   .fptr = open(filename, "r");
   if (!.fptr)
   {
      .Log ("Could not open file: ", filename);
   }
   else
   {
      .filename = filename;
      .Log ("File: ", filename, " opened");
      .ReadColumns();
   }
   .fptr;
}

method ReadCSV.CloseFile ()
{
   if (.fptr)
   {
      close (.fptr);
      .Log ("File: ", .filename, " closed");
      .fptr = nil;
   }
}

method ReadCSV.Trim(str)
{
   local  l = strlen(str), start, end;
   for (start=0; start<l && strchr(" \t",str[start]) != -1;)
      start++;
   for (end=l-1; end >= start && strchr(" \t",str[end]) != -1;)
      end--;
   if (start != 0 || end != 1-1)
      substr(str,start,end-start+1);
   else
      str;
}

method ReadCSV.ReadColumns ()
{  
local  line = read_line (.fptr);
local  i;

if (line != _eof_)
{
  line = list_to_array(string_split(line,.separators,
    0,t,"\"",t,"\",nil));
  for (i=0; i<length(line); i++)
  {
    if (.Trim(line[i]) == "")
    {
      line[i] = nil;
    }
else
    {
      if (strchr(line[i],':') == -1)
        line[i] = string(.domain,"":",line[i]);
      line[i] = symbol(line[i]);
      datahub_command(format("(create %s 1)",
        stringc(line[i])),I);
    }
  }
.columns = line;
.Log("Set columns to ", .columns);
}
}

method ReadCSV.GuessTypeValue (str)
{
local  value;
	ry
{
  value = parse_string(str,nil);
  if (!number_p(value))
    value = str;
}
catch
{
  value = str;
}
value;
}

method ReadCSV.ApplyLine (line)
{  
    local i, value;

    .Log ("Applying line: ", line);
    if (line)
    {
        line = list_to_array(string_split(line,.separators,
            0,t,"\\"",nil,"\\",nil));
        for (i=0; i<length(.columns); i++)
        {
            if (.columns[i])
            {
                value = .GuessTypeValue(line[i]);
                //.Log ("Set: ", .columns[i], " to ", stringc(value));
                if (value)
                    set(.columns[i], value);
            }
        }
    }
}

method ReadCSV.ReadLines ()
{
    local line, input;

    .Log ("Looking for new data...");
    while ((input = read_line(.fptr)) != _eof_)
    {
        if (.Trim(input) != ")
            line = input;
        }
    if (line)
    {
        .ApplyLine(line);
    }
}

method ReadCSV.ReadFile (filename)
{
    if (!.fptr)
        .OpenFile(filename);
    if (.fptr)
    {
        .ReadLines();
        if (.mode == #reload)
            .CloseFile();
    }
}
method ReadCSV.SetMode (mode) {
    .mode = mode;
    .Log("Set read mode to ", mode);
    .ChangeMenuItemLabel(.modemenu,
        string("Set ",
            (mode == #append) ? "Reload" :
            "Append",
            " Mode");
    if (.filename)
        .Reload(.filename);
}

method ReadCSV.ToggleMode () {
    .SetMode((.mode == #append) ? (#reload) : (#append));
}

method ReadCSV.ToggleVerbose () {
    .SetVerbose(!.verbose);
}

method ReadCSV.SetVerbose (mode) {
    .verbose = t;
    .Log("Set verbosity to ", (mode ? "verbose" : "quiet"));
    .verbose = mode;
    .ChangeMenuItemLabel(.verbosemenu,
        string(mode ? "Quiet" : "Verbose", " Mode");
}

method ReadCSV.Reload (filename) {
    .CloseFile();
    .ReadFile(filename);
}

/* Write the 'main line' of the program here. */
method ReadCSV.constructor () {
    .TimerEvery(.update_secs, `(@self).ReadFile((@self).filename));
.AddCustomSubMenu("CSV File Reader");
.AddCustomMenuItem("Reload CSV File",
  `(self).Reload((@self).filename));
.modemenu = .AddCustomMenuItem("Set Append Mode",
  `(self).ToggleMode());
.verbosemenu = .AddCustomMenuItem("Verbose",
  `(self).ToggleVerbose());
.SetVerbose(.verbose);
.SetMode(.mode);
}

method ReadCSV.ChangeMenuItemLabel (menuitemid, label)
{
  local parent = .CreateSystemMenu();
  local info = new MENUITEMINFO();

  if (cons_p(menuitemid))
    menuitemid = car(menuitemid);
  info.cbSize = 48;
  info.fMask = MIIM_STRING | MIIM_ID;
  info.fMask |= (WINVER < 0x0500 ? MIIM_TYPE : MIIM_FTYPE);
  info.fType = MFT_STRING;
  info.wID = menuitemid;
  info.dwTypeData = label;
  SetMenuItemInfo (parent, menuitemid, 0, info);
}

method ReadCSV.destructor ()
{
  .CloseFile();
}

ApplicationSingleton (ReadCSV);
WriteCSV.g

WriteCSV.g — writes data to CSV files.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called “How to Run a Script” for more information on using scripts.

```javascript
/*
 * Write data to a number of different CSV files. Each data set is
 * written at a different interval, either per second, per minute or
 * per hour. A new CSV * file is created every hour or every day.
 * 
 * The points names in each data set are read from a file. The file
 * consists of each point name, one per line. The output file will
 * contain a time stamp followed by the value for each point in the
 * order that the points are listed in the point name file.
 * 
 * To change the format of the file name, change the method
 * "GenerateFileName".
 * 
 * To change the extension of the file, change the member variable
 * "filesuffix".
 * 
 * To change the file names and timing, change the .NewDataSet calls
 * in the method WriteCSV.constructor.
 */

require ("Application");
require ("CSVSupport");

class WriteCSV Application
{
    datasets;
}

class MyCSVWriter CSVWriter
{
    pointfile;       // The name of a file containing the point list.
    sample_rate;     // one of #second, #minute, #hour or
```
// list(hour, minute, second)

file_rotate_rate; // one of #minute, #hour, #day or
// list(hour, minute, second)

timefile = ".csv"; // normally .csv. Override it to create
// .txt files.

} /*
 */

/*
 method MyCSVWriter.GenerateFileName()
 {
    local tm = localtime(nanoclock());
    format("%s_%d%02d%02d_%s%d%s", .filebase,
        tm.year+1900, tm.mon+1, tm.mday,
        tm.hour >= 12 ? "PM" : "AM",
        tm.hour > 12 ? tm.hour - 12 : (tm.hour == 0 ? 12 :
            tm.hour), .filesuffix);
    }
 */

/*
 method MyCSVWriter.GenerateFileName()
 {
    local tm = localtime(nanoclock());
    format("%s-%d%02d%02d-%02d%02d%s", .filebase,
        tm.year+1900, tm.mon+1, tm.mday,
        tm.hour, tm.min,
        .filesuffix);
    }
 */

/*
 * Create a new writer and start the timers to write new data and to
 * create a new log file.
 */

method WriteCSV.NewDataSet (output_file_base, pointfile, sample_rate,
    file_rotate_rate, separate_lines)
 {
    local writer = new MyCSVWriter();
    if (writer.ReadPointsFromCSV(pointfile))
    {
        writer.sample_rate = sample_rate;
    }
writer.file_rotate_rate = file_rotate_rate;
writer.SetFileBase(output_file_base);
writer.SetSeparateLines(separate_lines);

switch(writer.sample_rate)
{
  case (#second):
    TimerAt(nil,nil,nil,nil,nil,nil,
    `(@self).WriteData(@writer));
  case (#minute):
    TimerAt(nil,nil,nil,nil,nil,0,
    `(@self).WriteData(@writer));
  case (#hour):
    TimerAt(nil,nil,nil,nil,0,0,
    `(@self).WriteData(@writer));
  case (#day):
    TimerAt(nil,nil,nil,0,0,0,
    `(@self).WriteData(@writer));
  default:
    if (list_p(writer.sample_rate))
    {
      // List of hour, minute, second specification
      local  times = list_to_array(writer.sample_rate);
      TimerAt(nil,nil,nil,times[0],times[1],times[2],
      `(@self).WriteData(@writer));
    }
    else // Default is hourly
    {
      TimerAt(nil,nil,nil,nil,0,0,
      `(@self).WriteData(@writer));
    }
}
switch(writer.file_rotate_rate)
{
  case (#minute):
    TimerAt(nil,nil,nil,nil,nil,0,
    `(@self).RotateFile(@writer));
  case (#hour):
    TimerAt(nil,nil,nil,nil,0,0,
    `(@self).RotateFile(@writer));
  case (#day):
    TimerAt(nil,nil,nil,0,0,0,
    `(@self).RotateFile(@writer));
  default:
    if (list_p(writer.file_rotate_rate))
    {

// List of hour, minute, second specification
local times =
    list_to_array(writer.file_rotate_rate);
  .TimerAt(nil,nil,nil,times[0],times[1],times[2],
    `(self).RotateFile(@writer));
} else // Default is hourly
{
  .TimerAt(nil,nil,nil,nil,0,0,
    `(self).RotateFile(@writer));
}
.datasets = cons(writer, .datasets);
}

method WriteCSV.WriteData(writer)
{
  writer.WriteLine();
}

method WriteCSV.RotateFile(writer)
{
  .TimerAfter(0.1, `(self).IncrementFileName());
}

/* Write the 'main line' of the program here. */
method WriteCSV.constructor ()
{
  /* Specify the CSV files to write. Modify the .NewDataSet lines
     * to specify how to write each CSV file.
     * Arguments are:
     *   output_file_base:  The first part of the file, before the
     *                       date string
     *   pointfile:  The name of a file containing the point names
     *                for this data set
     *   sample_rate: One of #second, #minute, #hour or
     *                 list(hour,minute,second) telling how
     *                 frequently to write a line to the CSV file
     *   file_rotate_rate: One of #minute, #hour, #day
     *                    or list(hour,minute,second) telling how
     *                    frequently to create a new CSV file.
     *   separate_lines: If this is nil, write all point values on
     *                   one line. If this is t, write each point
     *                   value on a separate line.
     */
}
* This function will read the pointfile as a CSV file and treat
* the first field in each row as the name of a point to be
* recorded into the output file.
*
* When specifying timing, the input is the list of
* (hour,minute,second) as accepted by the "at" function
* (see documentation). A value of nil for hour, minute or
* second stands for all values for that parameter. A list of
* values for hour, minute or second specifies an event only
* when the hour, minute or second matches one of the values in
* the list.
* Examples:
*     list(nil,nil,nil)  - every second
*     list(nil,list(0,15,30,45),0) - every 15 minutes
*     list(nil,list(0,15,30,45),30) - every 15 minutes,
*                                  30 seconds past the minute
*     list(list(0,8,16), 0, 0)  - at midnight, 8AM and 4PM
*                                  exactly
*     list(list(0,8,16), 5, 0)  - at 12:05AM, 8:05AM and
*                                  4:05PM
*
*     The symbols #second, #minute, #hour, #day are conveniences for:
*     second:   list(nil,nil,nil)  - any hour, any minute,
*                every second
*     minute:   list(nil,nil,0)    - any hour, every minute
*                at 0 seconds
*     hour:     list(nil,0,0)     - every hour, on the hour
*     day:      list(0,0,0)       - at midnight (0 hour,
*                                  0 minute, 0 second)
*/
.NewDataSet("c:/tmp/Group1", "c:/tmp/Group1_Points.txt",
             #second, #hour, nil);
.NewDataSet("c:/tmp/Group2", "c:/tmp/Group2_Points.txt",
             #second, list(nil,list(0, 15,30,45),nil), nil);
.NewDataSet("c:/tmp/Group3", "c:/tmp/Group3_Points.txt",
             #second, list(nil,nil,0), nil);

/* Any code to be run when the program gets shut down. */
method WriteCSV.destructor ()
{
    with writer in .datasets do
    {
        destroy(writer);
    }
}
/* Start the program by instantiating the class. If your
* constructor code does not create a persistent reference to
* the instance (self), then it will be destroyed by the
* garbage collector soon after creation. If you do not want
* this to happen, assign the instance to a global variable, or
* create a static data member in your class to which you assign
* 'self' during the construction process. ApplicationSingleton()
* does this for you automatically. */
ApplicationSingleton (WriteCSV);
XMLReader.g

XMLReader.g — reads an XML file from a URL.

Code

The script allows you to process the data you receive in any way you like, and gives an example of printing the data with the Gamma `princ` function. If instead you would like to write the data to a point in the DataHub, you can replace this code:

```cpp
with point in model._xml_children do
{
    princ (point.name, " = ", point.value, "\n");
}
```

with this

```cpp
with point in model._xml_children do
{
    datahub_write (string("domain_name:",
        point.name), point.value);
}
```

where `domain` is the name of the domain in which you want the point to be created. Be sure to include the colon (:) at the end of the domain name, and make sure there are no other colons in the point name. For more information, please refer to the Data Domains section of the Cogent DataHub manual, and the `datahub_write` function in this manual.

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called "How to Run a Script" for more information on using scripts.

/*
This script reads an XML file from a URL and parses it into a class hierarchy that can be used to run further scripts or to populate the DataHub.

The example in this case requires that DataPid is running and that the DataHub web server is turned on, as the script is simply reading some data point values from the DataHub web server in XML format.
*/
require ("Application");

/* Get the Gamma library functions and methods for XML parsing and
   wget command-line tool */
require ("XMLSupport");
require ("WgetSupport");

class XMLReader Application
{
   // Note: The following line is wrapped for documentation formatting.
   // You will need to restore it to a single line to run the
   // program.
   TickSeconds = 5;
}

method XMLReader.loadXml ()
{
   // Call wget asynchronously. Protect against the possibility
   // that the script is stopped while a wget command is still
   // outstanding.
   Wget(.DocumentUrl, ‘progn {
      if (!destroyed_p(@self))
         (@self).processWgetResult();
   });
}

method XMLReader.processWgetResult ()
{
   if (ExitCode == 0) // success
   {
      // Print the raw XML document
      princ ("-------- Original Document --------\n");
      princ (ResultString);
      princ ("---------------------------------\n");

      // Parse the XML into a class hierarchy that we can easily
      // manipulate
      local reader =
         scew_reader_buffer_create(string_to_buffer(ResultString));
      local parser = new scew_parser();
      local tree = parser.load(reader);
      local model = tree.create_model("myxml_");

      // Print the whole XML file model.
// pretty_princ(model, "\n");

// The XML model consists of a class instance for each XML
// Element. Each class has a member called _xml_children
// that is an array of the sub-elements.
// In addition, each class has member variables that are
// named as the attributes of the XML element. Replace this
// with your own custom processing.
with point in model._xml_children do
{
    princ (point.name, " = ", point.value, "\n");
}
else
{
    princ ("XML read failed with exit code: ", ExitCode, "\n");
}

/* Write the 'main line' of the program here. */
method XMLReader.constructor ()
{
    .TimerEvery(.TickSeconds, `(@self).loadXml());
}

/* Any code to be run when the program gets shut down. */
method XMLReader.destructor ()
{
}

/* Start the program by instantiating the class. */
ApplicationSingleton (XMLReader);
ParseExcel.g

ParseExcel.g — parses data from an Excel spreadsheet.

Description

This script shows how to read an array of data into a DataHub script from a source such as an Excel worksheet, and then parse and extract values as necessary. The values in this example are message strings that correspond with an alarm, but they could be any value. The results of the triggered alarm are written to a text file.

Required setup

1. In the Excel worksheet:
   a. Select a group of cells in the worksheet that is 2 columns wide by any number of rows deep.
   b. Give this region a name by entering the name alarm_table in the Excel Name Box located in the top-left corner of the worksheet.
   c. Select a cell outside of this range, and name it time_stamp.

   This example assumes that there is a data feed to this time_stamp cell updating it with the most recent time. You could set this up to be coming from a DataHub point if desired, or from any other source.

2. In the DataHub's DDE properties:
   a. Add a new entry in the DDE Client section defined as follows:

   
   | Connection | alarms |
   | Name:      |       |
   | Service:   | Excel |
   | Topic:     | Alarms.xls (the name of the Excel file) |

   Enter the name alarm_table in the Item Names entry field, and choose Add. If the Data Domain column does not say default, double-click the name and change it to default.
   
   b. Now create another item named time_stamp in the same way.
   c. Select OK to close the DDE Item Definition window.
   d. Press Apply in the main properties window. The status of the DDE Connection should change to Connected.

3. In the DataHub Scripting properties:
   a. Select Add... to add a DataHub script.
   b. Navigate to the file ExcelAlarms.g, and choose Open. The script name should now appear in the list of scripts in the Scripting property page.
   c. Select the checkbox to the left of the ExcelAlarms.g file name, then press the
Apply button. This will cause the script to run whenever the DataHub starts.

d. With the ExcelAlarms.g file selected, press the Edit... button to open the file for editing in the Script Editor.

4. In the Script Editor:
   a. Go to line 35 and change the output file name to the name of a file that will receive the alarm log.
   b. Press the blue arrow icon or select Reload Whole File from the Script menu. The script is now running.

5. Close the Script Editor.

You can test the script by manually changing the alarm point values using the DataHub's Data Browser window. If you want to see output to the Script Log as well as the output file, un-comment lines 106, 107, 113, and 114.

This script assumes that the data for the alarm points comes from an outside data source connected to the DataHub via some other configuration.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called "How to Run a Script" for more information on using scripts.

```/* All user scripts should derive from the base "Application" class */

require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or Windows programming. Uncomment either or both. */

//require ("WindowsSupport");
//require ("ODBCSupport");

/* Applications share the execution thread and the global name space, so we create a class that contains all of the functions and variables for the application. This does two things:
  1) creates a private name space for the application, and
  2) allows you to re-load the application to create either a new unique instance or multiple instances without damaging an existing running instance. */```
/

class ParseExcel Application
{
    domain = "default";

    // The point containing the alarm table from Excel. This
    // is chosen from the DDE configuration tab in the OPC
    // DataHub properties.
    pt_alarm_table;
    pt_time_stamp;

    // The alarm lookup table. We parse the table we get from
    // Excel into a more efficient lookup table.
    alarm_lut = make_array(0);

    // The name of the log file.
    log_file_name = "c:/tmp/alarmlog.txt";

    // The file handle to the open file
    log_file;
}

/*
 * Hold information for one alarm entry in the Excel spreadsheet
 */
class AlarmSpec
{
    tagname;
    point;
    description;
    eventid;
}

method AlarmSpec.constructor (domain, tag, description)
{
    .tagname = tag;
    .description = description;
    .point = symbol(string(domain, ":", tag));
}

/*
 * Comparison function for sorting. We don't really need to sort
 * unless we plan to write code that looks up an alarm by name in
 * this table. It is more efficient to use the event handler
 * (.OnChange) function to map a point change to its alarm

/* specification */

function CmpAlarmSpecs (alarm1, alarm2)
{
    symcmp (alarm1.point, alarm2.point);
}

method ParseExcel.NewAlarmTable(value)
{
    local    rows = string_split (string(value), "\r\n", 0);
    local    columns;
    local    tagsym;

    with alarm in .alarm_lut do
    {
        .RemoveChange (alarm.eventid);
    }

    .alarm_lut = make_array(0);
    with row in rows do
    {
        columns = list_to_array (string_split (row, "\t", 0));
        .alarm_lut[length(.alarm_lut)] =
        new AlarmSpec(.domain, columns[0], columns[1]);
    }
    .alarm_lut = sort (.alarm_lut, CmpAlarmSpecs);

    with alarm in .alarm_lut do
    {
        datahub_command (string ("(create ",
            string(alarm.point), " 1")", 1));
        alarm.eventid = .OnChange (alarm.point, `(@self).
        AlarmOccurred
            (@alarm, value));
    }
}

/*
 * This method is called whenever the alarm condition point
 * changes in the OPC server.
 */

method ParseExcel.AlarmOccurred(alarm, value)
{
    if (value != 0)
    {
        writec (.log_file, format ("%-20s%-12s%s
",
$default:time_stamp,
  alarm.tagname, alarm.description));
  //princ (format ("%-20s%-12s%s\n", $default:time_stamp,
  //  alarm.tagname, alarm.description));
}
else
{
  writec (.log_file, format ("%-20s%-12s%s cleared\n",
    $default:time_stamp,
    alarm.tagname, alarm.description));
  //princ (format ("%-20s%-12s%s cleared\n", $default:time_stamp
  // alarm.tagname, alarm.description));
  }flush (.log_file);
}
/* Write the 'main line' of the program here. */
method ParseExcel.constructor ()
{
  .log_file = open (.log_file_name, "a");
  if (!.log_file)
  {
    MessageBox (0, string ("Could not open alarm log file: ",
      .log_file_name),
      "Error opening file", 0);
  }
  else
  {
    // Set the point name for the alarm table coming from Excel
    .pt_alarm_table = symbol (string
      (.domain, ":", "alarm_table"));
    .pt_time_stamp = symbol (string
      (.domain, ":", "time_stamp"));
    datahub_command (string ("(create ",
      stringc(.pt_alarm_table), " 1")"), 1);
    datahub_command (string ("(create ",
      stringc(.pt_time_stamp), " 1")"), 1);

    // Whenever somebody changes the Excel spreadsheet, update the
    // alarm table.
    .OnChange (.pt_alarm_table, `(self).NewAlarmTable(value));

    // If we already have a value for the alarm table point,
    // create the alarm table.
    if (!undefined_p(eval(.pt_alarm_table)) &&
string_p(eval(.pt_alarm_table))
    .NewAlarmTable (eval(.pt_alarm_table));
}

/* Any code to be run when the program gets shut down. */
method ParseExcel.destructor ()
{
    if (.log_file)
        close (.log_file);
}

/* Start the program by instantiating the class. If your
 * constructor code does not create a persistent reference to
 * the instance (self), then it will be destroyed by the
 * garbage collector soon after creation. If you do not want
 * this to happen, assign the instance to a global variable, or
 * create a static data member in your class to which you assign
 * 'self' during the construction process. ApplicationSingleton()
 * does this for you automatically. */
ApplicationSingleton (ParseExcel);
LinearXform.g

LinearXform.g — performs linear transformation functions on points.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\%

Please refer to the section called “How to Run a Script” for more information on using scripts.

```plaintext
/* This script creates a class that performs linear transformation functions on DataHub points. */
class LinearXform
{
    verbose;
}
method LinearXform.cbLinearXform (dst, value, mult, add)
{
    if (.verbose)
        princ ("xform ", dst, " = ", value, " * ", mult,
                " + ", add, "\n");
    if (number_p(value))
        set (dst, value * mult + add);
    else
        set (dst, value);
}
method LinearXform.AddLinearXform (app, src, dst, mult, add, bidirectional_p?=nil)
{
    datahub_command (string ("(create \"", src, "\" 1\")");
    datahub_command (string ("(create \"", dst, "\" 1\")");
    app.OnChange (src, `(@self).cbLinearXform (#@dst, value,
                                                @mult, @add));
    if (bidirectional_p && mult != 0)
    {
        allow_self_reference (src, 1);
        allow_self_reference (dst, 1);
        app.OnChange (dst, `(@self).cbLinearXform (#@src, value,
                                                @(1/mult),
                                                @(-add/mult)));
    }
}
```
MakeArray.g

MakeArray.g — creates an array point from individual points.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called "How to Run a Script" for more information on using scripts.

/*
 * This script creates an output array point from an input set of
 * individual points. The array can be any reasonable length, though
 * the algorithm is not efficient for large arrays whose constituent
 * points change quickly.
 * *
 * The only part of the code that you need to alter to create your
 * own arrays is the MakeArray.constructor method.
 * *
 * If a change is made to any individual point, the array will be
 * updated immediately.
 * *
 * If a change is made to the array point, the change will not affect
 * the individual points that make up the array.
 */

require ("Application");

/*
 * Applications share the execution thread and the global name
 * space, so we create a class that contains all of the functions
 * and variables for the application. This does two things:
 * 1) creates a private name space for the application, and
 * 2) allows you to re-load the application to create either
 * a new unique instance or multiple instances without
 * damaging an existing running instance.
 */
class MakeArray Application {

/* Ensure that a point exists in the DataHub data store. We do this
 * because the startup order of the DataHub vs. the data source is
 * not necessarily predictable. By creating the point now, we are

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
* sure that it exists even if the data source starts later.
*/
method MakeArray.CreatePoint(pointname, type?=nil)
{
    datahub_command(format("(create "%s" 1)", pointname), 1);
    if (type)
        datahub_command(format("(set_canonical "%s" "%s" 1)",
            pointname, type), 1);
}

/* Write the array based on the input point list. There are more
* efficient ways to do this if the array is large or the data
* updates very frequently. It may also be reasonable to do it on
* a timer rather than every time any constituent point changes.
*/
method MakeArray.EmitArray(arraypoint, inputpoints)
{
    local val = make_array(0), i=0, tmp;
    with point in inputpoints do
    {
        val[i++] = undefined_p(tmp =
            eval(symbol(point))) ? 0 : number(tmp);
    }
    set (symbol(arraypoint), val);
}

/* This is a convenient function that declares an array point to
* create from a set of input points. You can create as many of
* these array points as you need, using any number of input points.
* Just put all of the input points into the argument list of
* the call (see the call in the constructor below).
* The arraytype is a VARIANT array type: I1, I2, I4, R4, R8, BSTR,
* UI1, UI2, UI4 followed by a space and the word "array".
* E.g., "R8 array" or "BSTR array". AddCustomMenuItem
* BSTR means "string".
*/
method MakeArray.DeclareArray(arraypoint, arraytype, inputpoints...)
{
    .CreatePoint (arraypoint, arraytype);
    with point in inputpoints do
    {
        .CreatePoint(point);
        .OnChange(symbol(point),
            `(self).EmitArray(#@arraypoint, #@inputpoints));
    }
// After creating everything, call the EmitArray method once to
// initialize the array. We do this in case the constituent
// points are already present in the data set when the script
// starts. Otherwise we would have to wait until one
// of the points changes before the array gets initialized.
.EmitArray (arraypoint, inputpoints);

/* Write the 'main line' of the program here. Call the
 * .DeclareArray method one or more times to set up the event
 * handling to construct an array from individual points
 */
method MakeArray.constructor ()
{
   .DeclareArray ("default:pointarray", "R8 array",
                  "default:pointname1", "default:pointname2",
                  "default:pointname3");
}

/* Any code to be run when the program gets shut down. */
method MakeArray.destructor ()
{
}

/* Start the program by instantiating the class. If your
 * constructor code does not create a persistent reference to
 * the instance (self), then it will be destroyed by the
 * garbage collector soon after creation. If you do not want
 * this to happen, assign the instance to a global variable, or
 * create a static data member in your class to which you assign
 * 'self' during the construction process. ApplicationSingleton()
 * does this for you automatically. */
ApplicationSingleton (MakeArray);
BreakArray.g — breaks an array point into individual points.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\  

Please refer to the section called "How to Run a Script" for more information on using scripts.

```javascript
/*  
* Break an array point into individual points. For example, if we  
* have a point  
*   default:myarray = [ 1, 2, 3 ]  
* then we would like to create an output like  
*   default:myarray_0 = 1  
*   default:myarray_1 = 2  
*   default:myarray_2 = 3  
*  
* To set up an array to be broken into individual points,  
* call .MonitorArray(). This method takes a point name (of the  
* array) a format string and an index offset. The format string  
* specifies a suffix to add to the base point name, and the index  
* offset determines where to start numbering the suffix.  
* Typically index offset is 0 or 1.  
*  
* For example, to create points like:  
*   default:myarray_0  
*     use .MonitorArray("default:myarray", "_%d", 0)  
*   default:myarray_001  
*     use .MonitorArray("default:myarray", "_%03d", 1)  
*   default:myarray[1]  
*     use .MonitorArray("default:myarray", "[%d]", 1)  
*  
* This script will automatically respond to changes in the size of  
* the array by creating new points as the array expands, or by  
* marking existing points as not connected as the array contracts.  
*/

require ("Application");

class BreakArray Application
/* Write the 'main line' of the program here. You should only need
  * to modify the constructor to match your data points. */
method BreakArray.constructor ()
{
    // Delete the calls to .setupTest.
    // They are just here to test this script.
    .setupTest ("default:myarray", [ 1, 2, 3, 4, 5 ]);  
    .setupTest ("default:myarray2", [ 1, 2, 3, 4, 5 ]);

    // Add, remove or modify .MonitorArray calls to work with any
    // number of array points.
    .MonitorArray("default:myarray", "_%03d", 1);
    .MonitorArray("default:myarray2", "[%d]", 0);
}

/* ---------------------------------------------------------------- */
// You should not need to modify below this point.

/*
 * Create data points for an array using an index and a format string.
 * For example, to create points:
 *  default:myarray_0
 *     use suffixformat="_%d", indexoffset=0
 *  default:myarray[1]
 *     use suffixformat="[%d]", indexoffset=1
 *  default:myarray_001
 *     use suffixformat="_%03d", indexoffset=1
 */
method BreakArray.MonitorArray(pointname, suffixformat, indexoffset)
{
    local   value;
    .onChange(symbol(pointname), `(@(self).Break(this, value,
        @suffixformat,
        @indexoffset, t));

    // If we have a current value,
    // break the array for the first time now.
    if (!undefined_p(value = eval(symbol(pointname))))
    {
        .Break(symbol(pointname), value, suffixformat,
            indexoffset, nil);
    }
}
method BreakArray.Break(pointname, value, suffixformat, indexoffset, have_previous?=nil)
{
    local   elementname, elementsym, suffix;
    local   indx = indexoffset;
    local   info = PointMetadata(pointname), elementinfo;
    local   type, curlen, prevlen, i;

    //princ(info, "\n");

    if (array_p(value))
    {
        // Find the element type
        type = info.canonical_type & ~0xffffffff;

        // For each value in the array, create a point name for it.
        // If the point does not exist, or has an empty canonical
        // type, then create the point and set its canonical type to
        // the type of the parent array.
        with element in value do
        {
            suffix = format(suffixformat, indx);
            elementname = string(pointname, suffix);
            elementsym = symbol(elementname);
            elementinfo = PointMetadata(elementsym);
            if (!elementinfo || elementinfo.canonical_type == 0)
            {
                // This point has never been created in the DataHub
                // Create it and match its canonical type to the
                // array type.
                datahub_command (format("(create %s 1)",
                                         stringc(elementname)), 1);
                datahub_command (format("(set_canonical %s %d 1)"
                                         stringc(elementname),
                                         type), 1);
            }
        }

        datahub_write(elementname, element, nil,
                       info.quality, info.timestamp);
        indx++;
    }

    // Find the previous length. If the array used to be longer
    // then we should mark the values that are no longer present
// as Not Connected. If this function is called from within a
// change handler previous is implicitly defined.

if (have_previous)
{
    prevlen = (undefined_p(previous) ||
               !array_p(previous)) ? 0 : length(previous);
    curlen = (array_p(value) ? length(value) : 0);

    for (i=curlen; i<prevlen; i++)
    {
        suffix = format(suffixformat, indx);
        elementname = string(pointname, suffix);
        elementsym = symbol(elementname);
        elementinfo = PointMetadata(elementsym);
        if (elementinfo)
        {
            // princ ("Set array element ", elementname, " as not connected\n")
            datahub_write(elementname, elementinfo.value, nil,
                          OPC_QUALITY_NOT_CONNECTED, info.timestamp);
            indx++;  
        }
    }
}

/*/  
* This method is just used to create a test data set. 
*/
method BreakArray.setupTest(pointname, value)
{
    datahub_command (format("(create %s 1)", stringc(pointname)), 1);
    datahub_command (format("(set_canonical %s "R8 array" 1)", stringc(pointname)), 1);
    datahub_write (pointname, value);
}

/*/  
* Any code to be run when the program gets shut down. */
method BreakArray.destructor ()
{
}

/*/  
* Start the program by instantiating the class. */
ApplicationSingleton (BreakArray);
Example Scripts

IntToBit.g

IntToBit.g — converts an integer data point into a set of single-bit points.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\n
Please refer to the section called "How to Run a Script" for more information on using scripts.

```javascript
/*
 * Given an integer value, convert it to a set of data points where
 * each point represents a single bit in the original integer.
 *
 * Usage:
 * Modify the IntToBit.constructor to reflect the data points that
 * need to be broken into individual bits. This consists of one or
 * more calls to
 * .initPoint(point, format, nbits)
 * where
 * point - a symbol or string representing the input integer
 * data point specify symbols with #$domain:symbol_name
 * or as a string
 * format - a format string defining how to create the point
 * names for the individual bits, given the symbol name
 * and a bit number (starting at 0). Use %a for the
 * format specifier for input point name
 * nbits - the number of bits to extract from the input,
 * starting at the LSB
 */

require ("Application");

class IntToBit Application
{
} 

/* Add any data points that you want to split into bits here */
method IntToBit.constructor ()
{
    .initPoint(#$default:test, "%a_%d", 8);
    .initPoint("default:test2", "%a_%02d", 16);
}
```
Example Scripts

```plaintext
.initPoint(format("default:test%d", 3), "%a_%02d", 32);
}

/* ------- Implementation: No need to change beyond here ---------- */

/* A callback that runs whenever the input integer changes value */
method IntToBit.processNewValue (inputsym, outputformat, value, nbits)
{
    if (string_p(inputsym))
        inputsym = symbol(inputsym);

    local bitsym, bitvalue, i;
    local valueinfo = PointMetadata(inputsym);

    if (valueinfo)
    {
        for (i=0; i<nbits; i++)
        {
            bitsym = format(outputformat, inputsym, i);
            bitvalue = (value >> i) % 2;
            datahub_write(bitsym, bitvalue, nil, valueinfo.quality, valueinfo.timestamp);
        }
    }
}

/* Set up the event handler and initial state for the output points */
method IntToBit.initPoint (inputsym, outputformat, nbits)
{
    local bitsym, i, curvalue;

    if (string_p(inputsym))
        inputsym = symbol(inputsym);

    for (i=0; i<nbits; i++)
    {
        bitsym = format(outputformat, inputsym, i);
        datahub_command (format("(create %s 1)", stringc(bitsym)), 1);
        datahub_command (format("(set_canonical %s BOOL)"),
                       stringc(bitsym)), 1);
    }

    .OnChange(inputsym, `(@self).processNewValue(this, @outputformat, value, @nbits));

    if (!undefined_p(curvalue = eval(inputsym)) && number_p(curvalue))
    {
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
.processNewValue(inputsym, outputformat, curvalue, nbits);
}

/* Any code to be run when the program gets shut down. */
method IntToBit.destructor ()
{
}

/* Start the program by instantiating the class. */
ApplicationSingleton (IntToBit);
MaskedBridge.g

MaskedBridge.g — copies a data point, applying a mask and shift operation.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called “How to Run a Script” for more information on using scripts.

```plaintext
/*
 * Copy a data point to another data point, applying a mask and shift operation during the copy. The mask is applied before the shift, allowing you to mask any bits and then shift them right or left. A negative shift will shift left, and a positive shift will shift right. The shift indicates the number of bits to shift, where 0 indicates no shift.
 * You can create any number of operations simply by reproducing the .OnChange method call with different pairs of points. The datahub_command call is simply to ensure that the destination point exists, which may not be necessary depending on your application.
 */

require("Application");

class MaskedBridge Application
{
}

method MaskedBridge.WriteBits (value, pointname, mask, shift)
{
    if (shift < 0)
        set(pointname, (value & mask) << -shift);
    else
        set(pointname, (value & mask) >> shift);
}

method MaskedBridge.constructor ()
{
    datahub_command("(create DataPid:TwoBits 1)", 1);
}
```
.OnChange(#$DataPid:PID1.Mv,
   `(@self).WriteBits(value, #$DataPid:TwoBits, 0x03, 0));
}

/* Start the program by instantiating the class. */
ApplicationSingleton (MaskedBridge);
**ConnectionTrack.g**

ConnectionTrack.g — changes a point when a connection is made or broken.

**Code**

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

```
C:\Program Files\Cogent\Cogent DataHub\scripts\n```

Please refer to the section called "How to Run a Script" for more information on using scripts.

```javascript
/*
* This script watches the quality on an indicator point and sets an output to 0 if the quality is NOT CONNECTED and 1 otherwise.
* This effectively produces a synthetic point that changes according to whether a data connection to the source of the indicator point is made or broken.
*
* To use this script:
* 1) Adjust the poll_rate to the desired number of seconds. This can be fractional, such as 0.25.
* 2) Set track_time to t or nil. If set to t, the time stamp of the indicator point will be updated on each poll, causing a value change event to all attached clients, OPC servers, etc.
* 3) In the constructor, make one or more calls to .BeginTracking to set up a mapping between the point being watched and the indicator point. The watched_point
*
* Once this script is running, you can use the output point as a trigger to send email, write to a database, update a PLC, write to Excel or perform some other custom action through scripting.
*/

require ("Application");

class ConnectionTrack Application {
    poll_rate = 1; // polling rate in seconds
    track_time = nil; // set to t to adjust the output time stamp
        // on each poll
}

method ConnectionTrack.BeginTracking(indicator_point, output_point)
{
  datahub_command(format("(create %a 1)", indicator_point), 1);
  datahub_command(format("(create %a 1)", output_point), 1);
  .TimerEvery(.poll_rate, `(@self).CheckQuality(@indicator_point,
    @output_point));
}

method ConnectionTrack.CheckQuality(!indicator_point, !output_point)
{
  local quality = PointMetadata(indicator_point).quality;
  local active = 1;
  if (quality == OPC_QUALITY_NOT_CONNECTED)
    active = 0;
  datahub_write(string(output_point), active, .track_time);
}

method ConnectionTrack.constructor ()
{
  .BeginTracking(#$default:indicator, #$default:active);
  // default:indicator = the point you want to monitor
  // default:active = the point used to trigger the notification
}

ApplicationSingleton (ConnectionTrack);
QualityTrack.g

QualityTrack.g — writes the quality of a point as the value of another point.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called “How to Run a Script” for more information on using scripts.

/*
 * This script polls the quality of an "input" point and writes the
 * quality as the value of another "output" point.
 * The input point is typically an actual point from an OPC server.
 * The output point is a synthetic point that can be used to trigger
 * email and ODBC events.
 *
 * To alter this script, just edit the QualityTrack.constructor to
 * change the parameters of the .Track() call. The parameters are:
 * poll_secs - the polling rate. This can be fractional, as in 0.5
 * input - the name of the input point as a symbol
 * output - the name of the output point as a symbol
 *
 * You can add as many .Track() calls as you need to monitor
 * multiple points.
 *
 * In an email or ODBC Condition configuration, the quality of a
 * point can be compared to the OPC quality constants:
 * OPC_QUALITY_BAD
 * OPC_QUALITY_COMM_FAILURE
 * OPC_QUALITY_CONFIG_ERROR
 * OPC_QUALITY_DEVICE_FAILURE
 * OPC_QUALITY_EGU_EXCEEDED
 * OPC_QUALITY_GOOD
 * OPC_QUALITY_LAST_KNOWN
 * OPC_QUALITY_LAST_USABLE
 * OPC_QUALITY_LOCAL_OVERRIDE
 * OPC_QUALITY_MASK
 * OPC_QUALITY_NOT_CONNECTED
 * OPC_QUALITY_OUT_OF_SERVICE
 * OPC_QUALITY_SENSOR_CAL
 * OPC_QUALITY_SENSOR_FAILURE
require ("Application");
require ("Time");

class QualityTrack Application
{
}

method QualityTrack.CheckQuality(input, output)
{
    local info = PointMetadata(input);
    set (output, info.quality);
}

method QualityTrack.Track(poll_secs, input, output)
{
    datahub_command(format("(create %a 1)", output));
    .TimerEvery(poll_secs, `(@self).CheckQuality(#@input, #@output));
}

method QualityTrack.constructor ()
{
    // Change the poll_secs and the two point names here.
    // The # in front of the point name is necessary.
    .Track(1, #$default:my_real_opc_point,
         #$default:my_synthetic_trigger);

    // Add more calls to .Track() here
}

ApplicationSingleton (QualityTrack);
TagMonitor.g

TagMonitor.g — monitors DataHub points for changes in quality or failure to change value.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\%

Please refer to the section called "How to Run a Script" for more information on using scripts.

```javascript
/*
 * This script monitors tags (or DataHub points) for two conditions:
 * 1) change of quality
 * 2) failure to change within a time period
 * For each of these, the script creates synthetic points
 * (the "target") that hold the result of monitoring the watch points.
 * Email and database events can then be triggered from the
 * synthetic points.
 * You can modify the constructor function in this script to change
 * the point names, or to add additional watch conditions as needed.
 */
require ("Application");

class TagMonitor Application {

    /*
     * Set up a watch tag and a target tag such that the quality of the
     * watch tag is copied into the value of the target tag.
     */
    method TagMonitor.copyQuality(poll_seconds, watch, target) {
        // Ensure that the input and output tags exist.
        datahub_command(format("(create %s 1)", stringc(watch)), 1);
        datahub_command(format("(create %s 1)", stringc(target)), 1);
    
    }
}
Example Scripts

// Periodically copy the quality of the watch point into
// the value of the target
TimerEvery(poll_seconds,
`set(#@target, PointMetadata(#@watch).quality));
}

/*
* Set up a watch tag, a target tag and a time such that the target
* tag will be set to 1 if the watch tag has changed within the timer
* period, or zero if the watch tag has not changed within the time
* period. The time period is specified by dead_seconds, which may
* be fractional.
*/
method TagMonitor.copyChangeStatus(dead_seconds, watch, target)
{
    // Ensure that the input and output tags exist.
datahub_command(format("(create %s 1)", stringc(watch)), 1);
datahub_command(format("(create %s 1)", stringc(target)), 1);

    // Start the watch point off as having changed
setprop(watch, #has_changed, t);

    // Whenever the watch point changes, set its property
    // "has_changed" to t.
.OnChange(watch, `(self).watchHasChanged(#@watch, #@target));
.TimerEvery(dead_seconds,
    `(self).checkChange(#@watch, #@target));
}

/*
* A callback function that checks for a change in a point and puts
* a 1 or 0 into the target. Reset the changed flag to zero.
*/
method TagMonitor.checkChange(watch, target)
{
    set(target, getprop(watch, #has_changed) ? 1 : 0);
    setprop(watch, #has_changed, nil);
}

/*
* A callback whenever a change watch point changes. We use this to
* change from 0 to 1 as soon as we see a change in a watch point
* instead of waiting for the poll delay.
*/
method TagMonitor.watchHasChanged(watch, target)
{
setprop(watch, #has_changed, t);
set(target, 1);

/* Write the 'main line' of the program here.
*
* As written, the points to watch for quality and change, as well as
* the target points to modify, are all in the "default" domain,
* as follows:
*
* Point to watch for quality:  default:quality_watch
* Point to watch for change:   default:change_watch
* Target point for quality:    default:quality_target
* Target point for change:     default:change_target
*
* You can change these to different domain and point names. Also,
* you can add any number of other points to monitor quality and
* change, following the same syntax.
*
* The first argument of .copyQuality is the poll rate in seconds on
* the quality of the point. The first argument of .copyChangeStatus
* is the number of "dead" seconds to wait for a change, before
* notifying of a failure.
*/
method TagMonitor.constructor ()
{
    .copyQuality(1, #$default:quality_watch,
                #$default:quality_target);
    .copyChangeStatus(5, #$default:change_watch,
                      #$default:change_target);
}

/* Any code to be run when the program gets shut down. */
method TagMonitor.destructor ()
{
}

/* Start the program by instantiating the class. If your
* constructor code does not create a persistent reference to
* the instance (self), then it will be destroyed by the
* garbage collector soon after creation. If you do not want
* this to happen, assign the instance to a global variable, or
* create a static data member in your class to which you assign
* 'self' during the construction process. ApplicationSingleton()
* does this for you automatically. */
ApplicationSingleton (TagMonitor);
TimedUpdate.g

TimedUpdate.g — periodically updates the timestamp on a set of DataHub points without changing their values.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called “How to Run a Script” for more information on using scripts.

```/*
 * This script runs a timer that periodically updates the timestamp on
 * a set of data points without changing their values.
 */
require ("Application");

/*
 * Modify the list of point names here to change which points will be
 * written. Modify the update time (in seconds) to change the write
 * frequency
 */
class TimedUpdate Application
{
    points = [ "default:test1", "default:test2" ];
    updateSecs = 5;
}

/* This is the callback that runs when the timer fires */
method TimedUpdate.doUpdate ()
{
    local current;
    with point in .points do
    {
        current = datahub_read(point);
        if (current[0])
        {
            // preserve the current value and quality,
            // but let the time change to the current
            // system clock time by not specifying
```
// the time argument. The "1" for the "force"
  // argument indicates that the DataHub should
  // emit a change even if the settings would
  // normally cause this change to be ignored.
  datahub_write(point, current[0].value, 1,
                  current[0].quality);

// Write the 'main line' of the program here.
// This is where we start the timer. */
method TimedUpdate.constructor ()
{
  // The .TimerEvery function will start counting when the
  // script starts running, so it will not be synchronized with
  // a particular time of day.
  .TimerEvery (.updateSecs, `(@self).doUpdate());

  // If you want to synchronize the update with the time of day,
  // say exactly on each half-hour, you would use the .TimerAt
  // function. The arguments are .TimerAt(day, month, year,
  // hour, minute, second, callback_function)
  // Specify nil to mean "any". Specify a list to give
  // multiple values. For example, this will perform the update
  // at (0 and 30 minutes) and 0 seconds past every hour:
  // .TimerAt (nil, nil, nil, nil, list(0, 30), 0,
  //          `(@self).doUpdate());

  /* Any code to be run when the program gets shut down. */
  method TimedUpdate.destructor ()
  {
  }

  /* Start the program by instantiating the class. */
  ApplicationSingleton (TimedUpdate);
FixQuality.g

FixQuality.g — changes point quality for OPC clients that treat bad quality as a disconnection.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called "How to Run a Script" for more information on using scripts.

/* This application monitors a set of points in the "input domain" and copies them to the "output domain". If the quality of the input point is not GOOD, then the script modifies the point value to -1 and the quality to GOOD. This is to handle an OPC client that treats bad quality as a disconnection.
 *
 * To configure this application, modify the class variables:
 *    domain_in = the name of the input domain
 *    domain_out = the name of the output domain
 *    value_if_bad = the value to substitute on the point if the quality is bad. If this is nil, then do no value substitution.
 */
require ("Application");

class FixQuality Application
{
    domain_in = "test";
    domain_out = "test2";
    value_if_bad = -1;
}

/* Monitor a point. This includes creating the point if it does not exist, and then creating the output domain's mirror of the point. This function also sets up an event handler to map any future changes of the point into the output domain. */
method FixQuality.Monitor (ptname)
{
    local outname, ptsym;

    outname = string(.domain_out, ":", ptname);


Example Scripts

```plaintext
ptname = string(.domain_in, ":", ptname);
.OnChange(symbol(ptname),
   `(@self).BridgeQuality(#@ptname, #@outname));
datahub_command(format("(create %s 1)", stringc(ptname)), 1);
ptsym = symbol(ptname);
if (!undefined_p(eval(ptsym))
   .BridgeQuality(ptname, outname);
}

/* This is the function that does the work of mapping from the
input domain to the output domain. */
method FixQuality.BridgeQuality(ptname, outname)
{
   local  info = PointMetadata(symbol(ptname));
   local  value = info.value;
   if (info.quality != OPC_QUALITY_GOOD && .value_if_bad)
      value = .value_if_bad;
   datahub_write(outname, value, 1,
      OPC_QUALITY_GOOD, info.timestamp);
}

method FixQuality.MonitorDomain(domain)
{
   local  points = datahub_points(domain, nil);
   with point in points do
      {
         .Monitor(point.name);
      }
}

/* This is the mainline of the program. You can either call
 .Monitor("pointname") for each point, or you can call
 .MonitorDomain(.domain_in) to monitor all existing points in
 the input domain. If you choose to monitor the whole domain,
you must re-run this application whenever new points are added
to the domain. */
method FixQuality.constructor ()
{
   .Monitor("point001");
   .Monitor("point002");
   .MonitorDomain(.domain_in);
}

/* Any code to be run when the program gets shut down. */
method FixQuality.destructor ()
{

```
/ * Start the program by instantiating the class. If your
 * constructor code does not create a persistent reference to
 * the instance (self), then it will be destroyed by the
 * garbage collector soon after creation. If you do not want
 * this to happen, assign the instance to a global variable, or
 * create a static data member in your class to which you assign
 * 'self' during the construction process. ApplicationSingleton()
 * does this for you automatically. */
ApplicationSingleton (FixQuality);
Example Scripts

OPCItemLoader.g

OPCItemLoader.g — reads a list of OPC DA tags from a CSV file and configures DataHub points for them.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\

Please refer to the section called "How to Run a Script" for more information on using scripts.

```/* This script reads a set of OPC item names and point names from a
 * CSV file and updates the Manually Selected Items configuration for
 * an OPC connection. The CSV file format can be one of:
 *  OPC_ITEM_ID
 * or
 *  OPC_ITEM_ID, OPC_DATAHUB_POINT_NAME
 *
 * If the OPC_DATAHUB_POINT_NAME is absent (the line contains only an
 * OPC item ID), then the script will create an OPC DataHub point
 * with the same name as the OPC item ID. The OPC item ID is the
 * item ID defined by the OPC server.
 *
 * To use this script, follow these steps:
 *
 * 1) Create the OPC connection that you want to add items to.
 * Select "Manually Select Items" only in the "Item Selection"
 * section. You do not need to configure any items. Press
 * OK to save the OPC configuration, the press Apply on the
 * DataHub properties dialog.
 * 2) Open this script in the editor and change the parameters in the
 * OPCItemLoader class definition. These are documented below.
 * 3) Close the OPC DataHub Properties window.
 * 4) Run this script by pressing the run button in the toolbar of
 * the editor (the right-facing blue arrow).
 * 5) Open the OPC DataHub properties dialog, open the OPC
 * configuration for your server and verify that the items were
 * added.
 *
 * If the script has problems, you should see error messages in the
 * "Script Log" window. */```
* You must close the OPC DataHub Properties window before running
* this script or the changes made by this script may be lost.

* You do not need to run this script each time the DataHub is
* started. The configuration produced by this script will be saved
* in the DataHub's configuration file permanently.

* Editable fields:
*   connection_name = the name of the OPC connection to be adjusted.
*       This is the name entered into the box marked "Connection
*       Name:" in the "Define OPC Server" dialog of the DataHub
*       Properties dialog.
*   file_name = the file name containing the OPC item names and
*       point names. The file can be either one or two strings per
*       line separated by commas. If only a single string appears,
*       it is taken to be the OPC server item name. The point name
*       is computed from the item name. If two strings appear then
*       the first string is the OPC item and the second string is
*       the DataHub point name. The point name will automatically
*       be broken into components on a "." and a tree hierarchy
*       will be created as if each component but the last is a tree
*       branch.
*   trim_spaces = t if you want the item and point names to be
*       trimmed of all leading and trailing white space and tabs,
*       otherwise nil.
*   path_separator = nil if you do not want to split point names
*       to produce a tree hierarchy, otherwise a string containing
*       separator characters. Normally this will be a "." character.

```javascript
require ("Application");
require ("WindowsSupport");
require ("OPCSupport");

class OPCItemLoader Application
{
    connection_name = "OPC000";
    // Change this to the connection name of the connection to edit
    file_name = "my_items.csv";
    // Change this to the file containing the item and point names
    trim_spaces = t;
    // set to nil to preserve leading and trailing spaces in item
    // names
    path_separator = ".";
    // Characters used to split point names into tree components
    file_is_8bit_ansi = t;
}```
// Set to t if file uses 8-bit extended ANSI, nil if 7-bit ASCII
// or UTF-8
opc_connection;

method OPCItemLoader.Trim (str)
{
    local i=0, j, len;
    len = strlen(str);
    while (i < len && (str[i] == ' ' || str[i] == '	'))
        i++;
    j = len - 1;
    while (j >= i && (str[j] == ' ' || str[j] == '	'))
        j--;
    if (j>=i)
        str = substr(str, i, j-i+1);
    else
        str = "";
    str;
}

method OPCItemLoader.ReadCSVFile (filename)
{
    local fptr = open (filename, "r", nil);
    local line, i;
    if (fptr)
    {
        while ((line = read_line(fptr)) != _eof_)
        {
            if (.file_is_8bit_ansi)
                line = strcvt(line);
            if (line != "")
            {
                line = list_to_array(string_split(line, ",", 0, nil, nil, nil, nil, "\\", nil));
                if (.trim_spaces)
                {
                    for (i=0; i<length(line); i++)
                    {
                        line[i] = .Trim(line[i]);
                    }
                }
            }
        }
        close (fptr);
    }
}
if (length(line) == 1)
   .AddOPCItem(line[0], line[0]);
else if (length(line) > 1)
   .AddOPCItem(line[0], line[1]);
}
}
close (fptr);
}
else
{
   local s = strerror(errno());
   prin(class_name(self), ": Could not open file: ",
   filename, ": ", s, "\n");
}

method OPCItemLoader.AddOPCItem(itemname, pointname)
{
   .opc_connection.addItem(pointname, itemname, OPC_NODE_LEAF,
   .path_separator);
}

method OPCItemLoader.LoadFromCSV(opc_conn_name, filename)
{
   local opc = new OPCConnection();
   opc.setServer(opc_conn_name);
   .opc_connection = opc;
   .ReadCSVFile(filename);
}

/* Write the 'main line' of the program here. */
method OPCItemLoader.constructor ()
{
   .LoadFromCSV(.connection_name, .file_name);
}

/* Any code to be run when the program gets shut down. */
method OPCItemLoader.destructor ()
{
}

ApplicationSingleton (OPCItemLoader);
**OPCReconnect.g**

OPCReconnect.g — disconnects and reconnects an OPC server.

**Code**

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

```
C:\Program Files\Cogent\Cogent DataHub\scripts\
```

Please refer to the section called "How to Run a Script" for more information on using scripts.

```/* Sample script that disables the OPC connection labeled "OPC000",
  * waits 2 seconds, then re-enables the connection. The result will
  * be that the DataHub disconnects from the server for 2 seconds
  * and then reconnects.
  */
require ("Application");
require ("OPCSupport");

class OPCReconnect Application
{

  /* Create an object that references an existing OPC connection that
   * we have configured through the user interface. Its label is the
   * first argument to setServer below. Once we have the OPC connection
   * object, we can call enable() with t or nil in the argument to
   * enable or disable this connection.
   */
  method OPCReconnect.enable(enabled)
  {
    local opcclient = new OPCConnection();
    opcclient.setVerbose(t);
    opcclient.setServer("OPC000");
    opcclient.enable(enabled);
  }

  method OPCReconnect.constructor ()
  {
    .enable(nil);
    .TimerAfter(2, `(@self).enable(t));
  }
```
ApplicationSingleton (OPCReconnect);
OPCReload.g

OPCReload.g — requests a reload of OPC server data with no disconnect.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\n
Please refer to the section called “How to Run a Script” for more information on using scripts.

/* This script requests a reload of data from a connected OPC server without disconnecting from the server. It is particularly useful for automatically adding new points, and can poll for new points based on a timer or on a trigger point.

* You will need to edit the following three variables to run the script on your system:

* connection_label: The name given to your OPC connection when you defined your OPC server. This appears in the "Connection" column in your list of OPC Client connections in the Properties window.

* poll_time: The poll rate to check for new points, in seconds.

* trigger_point: The name of the point you will use as a trigger.

*/

require ("Application");
require ("OPCSupport");

class OPCReload Application {
    connection_label = "OPC002"; // connection label in "Connection" column of OPC properties
    poll_time = 30; // poll rate for new points, in seconds
    trigger_point = #$domain_name:point_name; // name of point used as a trigger
}
/* Transmit a "reload" command to the server, without disconnecting. 
* To disconnect and reload, change the parameter to opc.reload to t
* instead of nil. */

method OPCReload.reload ()
{
    local opc = new OPCConnection();
    opc.setVerbose(t);
    opc.setServer(.connection_label);
    opc.reload(nil);
}

/* Start the polling timer */
method OPCReload.constructor ()
{
    // To poll for new points periodically, use this line
    .TimerEvery (.poll_time, `(@self).reload());

    // To poll for new points only when a specific data point changes,
    // use this line
    .OnChange (.trigger_point, `(@self).reload());
}

/* Instantiate the class. This calls the constructor. */
ApplicationSingleton (OPCReload);
**AutoCalculation.g**

AutoCalculation.g — automatically calculates formulas based on data points.

**Code**

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

```
C:\Program Files\Cogent\Cogent DataHub\scripts\
```

Please refer to the section called "How to Run a Script" for more information on using scripts.

```gamma
/*
 * Automatically calculate formulas based on data points. Create data points where necessary to store the results back to the DataHub's data set.

 * To use:
 * - create a file, like c:/tmp/calculations.txt
 * - in the file, create any number of calculations of the form

 * $default:test = $DataPid:PID1.Mv * 10;
 * $default:test2 = $DataPid:PID1.Mv / 10;
 * - change the fileName member in the AutoCalculation class below
 * - run the script

 * Any symbol in the calculation of the form $domain:name is assumed to be a DataHub point, and will be created if necessary. The point on the left hand side of the equal sign will be automatically updated whenever any point on the right hand side of the equal sign changes.

 * You can include complex Gamma expressions, like:

 * $default:output = progn {
 *     local a = $DataPid:PID1.Mv;
 *     local b = $DataPid:PID1.Pv;
 *     if (b != 0)
 *         a / b;
 *     else
 *         a;
 * }

 * You can define functions within this file, though it is better to put functions into a separate script file:
```
function average(a,b) {
    a + b / 2;
}

$default:output = average($DataPid:PID1.Mv, $DataPid:PID1.Pv);

require ("Application");

class AutoCalculation Application {
    fileName = "C:/tmp/calculations.txt";
    fp;
}

class Computation {
    app; // the Application instance defining this computation
    output; // a symbol
    inputs; // a list of symbols
    code; // the code to execute
}

method Computation.constructor(app, expression) {
    .app = app;
    .code = expression;
    .output = .findOutput(expression);
    .inputs = .findInputs(expression);

    if (.output)
    {
        datahub_command(format("(create %s 1)", stringc(.output)), 1);
        with input in .inputs do
        {
            datahub_command(format("(create %s 1)", stringc(input)), 1);
            .app.OnChange(input, `(@self).compute());
        }
    }
    .compute();
}
method Computation.compute()
{
    try
    {
        eval(.code);
    }
    catch
    {
        princ("Assignment to ", .output, " failed: ",
            _last_error_, "\n");
    }
}

method Computation.findOutput (expr)
{
    if (car(expr) == '#setq)
        cadr(expr);
    else
        nil;
}

method Computation.findInputsRecursive (expr)
{
    local inputs, partial;
    if (list_p(expr))
    {
        with part in expr do
        {
            partial = .findInputsRecursive(part);
            inputs = nappend(inputs, partial);
        }
    }
    else if (symbol_p(expr) && strchr(string(expr), ":") != -1)
    {
        inputs = cons(expr, inputs);
    }
    inputs;
}

method Computation.findInputs (expr)
{
    .findInputsRecursive(cddr(expr));
}

method AutoCalculation.readFile()
{
if ((.fp = open(.fileName, "r", t)) != nil)
{
    //create some local variables
    local line;

    //loop until we have read the entire file.
    while((line = read(.fp)) != _eof_)
    {
        local comp = new Computation(self, line);
    }

    //once we have read the entire file we need to close it.
    close(.fp);
}
else
{
    princ ("AutoCalculation: Could not open file: ",
    .fileName, ": ", strerror(errno()), ": ", "\n");
}

/* Write the 'main line' of the program here. */
method AutoCalculation.constructor ()
{
    .readFile();
}

/* Any code to be run when the program gets shut down. */
method AutoCalculation.destructor ()
{
}

/* Start the program by instantiating the class. */
ApplicationSingleton (AutoCalculation);
Built-in Classes
**DH_Domain**

DH_Domain — the structure of a DataHub domain.

**Synopsis**

```cpp
class DH_Domain
{
    name; // the name of the domain, as a string
    auto_created; // 1 if the domain was created implicitly due to
                   // a point reference,
                   // or 0 if the domain was configured in the
                   // General tab
    n_points; // the number of points in this domain, including
               // points with values and branches in the
               // hierarchy tree
    n_assemblies; // Not implemented
    n_attributes; // Not implemented
    n_bridges; // The number of domain bridges or redundancy pairs
               // in which this domain is participating
}
```
**DH_Item**

DH_Item — the structure of a DataHub point.

**Synopsis**

class DH_Item
{
    canonical_type;     // number: The canonical VARIANT type
    conf;               // number: confidence (0-100)
    domain;             // string: The domain name
    flags;              // number: flags describing the point
    n_aliases;          // number of aliases for this point
    n_attributes;       // number of attributes this point has
    n_pending;          // number of clients with a change pending
    n_properties;       // number of properties this point has
    n_registered;       // number of clients registered for changes
    n_subassemblies;    // number of subassemblies this point has
    name;               // string: name of the point, without
                       // the domain
    opcaccessrights;    // number: read/write flags
    propid;             // number: property id if applicable
    quality;            // number: OPC item quality
    security;           // number: security level (not used)
    timestamp;          // number: time stamp in Windows epoch time
    value;              // variant: the point value
}

**Instance Variables**

flags

This instance variable may be a combination of zero or more of the following flags.

<table>
<thead>
<tr>
<th>Hex</th>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00001</td>
<td>READABLE</td>
<td>Item is readable.</td>
</tr>
<tr>
<td>0x00002</td>
<td>WRITABLE</td>
<td>Item is writable.</td>
</tr>
<tr>
<td>0x00004</td>
<td>LOCKED</td>
<td>Item is locked (not used).</td>
</tr>
<tr>
<td>0x00008</td>
<td>PROPERTY</td>
<td>Item is a property of another point.</td>
</tr>
<tr>
<td>0x00010</td>
<td>SUBASSEMBLY</td>
<td>Item is a subassembly of another point.</td>
</tr>
<tr>
<td>0x00020</td>
<td>ASSEMBLY</td>
<td>Item is an assembly.</td>
</tr>
<tr>
<td>0x00040</td>
<td>ATTRIBUTE</td>
<td>Item is an attribute of another point.</td>
</tr>
<tr>
<td>0x00080</td>
<td>TYPE</td>
<td>Item is an attribute type.</td>
</tr>
<tr>
<td>Hex</td>
<td>Constant</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x00100</td>
<td>ACTIVE</td>
<td>Item is active.</td>
</tr>
<tr>
<td>0x00200</td>
<td>PRIVATE_ATTRIBUTE</td>
<td>Item is a private attribute of a type.</td>
</tr>
<tr>
<td>0x00800</td>
<td>HIDDEN</td>
<td>Item is hidden.</td>
</tr>
<tr>
<td>0x01000</td>
<td>AUTO_ID</td>
<td>Item was assigned a <em>propid</em> automatically.</td>
</tr>
<tr>
<td>0x40000</td>
<td>TEMP_VALUE</td>
<td>Item's value is temporary, probably Not Connected assigned by an interface becoming disconnected.</td>
</tr>
</tbody>
</table>
Special Gamma Functions for DataHub Scripting
_ — looks up a translation text.

Synopsis

_ (string)

Arguments

string
A string that should be translated.

Returns

The translation of the string.

Description

This function looks up a pre-configured translation text. It is used internally by the DataHub interface programmer to specify which text strings in the interface should be translated when the user selects a language in the Properties window of the DataHub. Language selection itself cannot be done within Gamma; it can only be done through the Properties window.

If your language is not available in the DataHub, you can contact Cogent for instructions on how to add a translation to the DataHub source code.

The unusual name for this function is based on a convention of GNU's gettext function, in which the underscore symbol (_) can be used in place of the gettext function name.
add_menu_action

add_menu_action — adds a menu action.

Synopsis

add_menu_action (menu_id, s_exp)

Arguments

menu_id
A unique ID number for this item.

s_exp
A Gamma expression that is the action to be taken when a menu item is selected.

Returns

A list:

(menu_id s_exp)

Where the s_exp is in Lisp format.

Description

This function lets you specify what code gets run when a menu item is selected. The code can be removed using remove_menu_action.

Example

This example is part of the WindowsExample.g example program:

```gamma
method WindowsExample.AddSubMenu (parent, pos, label)
{
  local submenu = CreatePopupMenu();
  InsertMenu (traymenu, pos, MF_BYPOSITION | MF_POPUP,
    submenu, label);
  .menu_items = cons (cons (submenu, t), .menu_items);
}

method WindowsExample.AddMenuItem (parent, pos, label, code)
{
  local info = new MENUITEMINFO();
  info.cbSize = 48;
```
```lua
info.fMask = MIIM_STRING | MIIM_FTYPE | MIIM_ID;
info.fType = MFT_STRING;
info.wID = ++MenuItemID;
info.dwTypeData = label;
InsertMenuItem (parent, pos, 1, info);
local action = add_menu_action (MenuItemID, code);
.menu_actions = cons (action, .menu_actions);
}

method WindowsExample.AddMenus ()
{
  local  traymenu = get_tray_menu ();

  if (traymenu != 0)
  {
    local submenu = .AddSubMenu (traymenu, 5, "Monitor Functions");

      .AddMenuItem (submenu, -1, "Select Color",
        `(self).SelectTrayMenuItem (@MenuItemID+1));
    .AddMenuItem (submenu, -1, "Select File",
      `new GFileDialog (1).DoModal(0));
    .AddMenuItem (submenu, -1, "Select Folder",
      `new GFolderDialog ().DoModal(0));
  }
else
  {
  }
}
allow_self_reference

allow_self_reference — permits changes to be written back to the point of origin.

Synopsis

allow_self_reference (symbol, allow)

Arguments

symbol

Any valid Gamma symbol.

allow

1 or any other non-zero value allows self-referential behavior; 0 disallows it.

Returns

The value of the allow parameter.

Description

This function tells Gamma not to generate a warning if a change function like on_change causes a sequence of events that changes the original point again.

Example

This example is from LinearXform.g.

```pseudo
method LinearXform.AddLinearXform (app, src, dst, mult, add, bidirectional_p?=nil)
{
    app.OnChange (src, `(self).cbLinearXform (#dst, value, @mult, @add));
    if (bidirectional_p && mult != 0)
    {
        allow_self_reference (src, 1);
        allow_self_reference (dst, 1);
        app.OnChange (dst, `(self).cbLinearXform (#src, value,
            1/@mult,
            (@-add)/(@mult)));
    }
}
**datahub_command**

`datahub_command` — sends commands to the DataHub.

**Synopsis**

```
datahub_command (command, sync?)
```

**Arguments**

- `command`  
  A string containing a DataHub configuration command.

- `sync`  
  1 sends the command synchronously; 0 sends the command asynchronously.

**Returns**

The result of the command, as a string.

**Description**

This function sends configuration commands to the DataHub, in Lisp format. It returns the result of the command as a string. For more information on DataHub commands, please refer to the Using DataHub Commands chapter of the *Cogent DataHub* manual.

Setting the `sync` parameter to 1 (synchronous) may slow the execution a tiny bit, but it ensures that the DataHub will complete the command before your script continues. Setting the `sync` parameter to 0 will allow your script to continue whether or not the DataHub command has executed. For most cases, we recommend setting it to 1, just to be on the safe side. This ensures that your script will execute commands in the order expected. The tiny delay for synchronous execution is in micro-milliseconds—insignificant for most applications.

**Escaping Characters**

Since commands are sent within strings, you probably need to escape certain characters, using the backslash (\). For example, to escape the quote marks delineating the string "my_string", you would enter the string as: "my_string".

**Windows File Paths**

Normally if a command is in a configuration file, you need to escape the backslash in file names, like this:

```
c:\tmp\datahub.log
```
But, if you are issuing the command from gamma using datahub_command you need to escape each of those two backslashes, like this:

```
c:\\tmp\\datahub.log
```

So the command would be:

```
datahub_command ("(log_file "c:\\tmp\\datahub.log\")")
```

Since the quotes are optional (though recommended) in DataHub commands you could do this:

```
datahub_command ("(log_file c:\\tmp\\datahub.log")
```

Fortunately, recent Microsoft operating systems will accept forward slashes in file names, so for those versions you can do this:

```
datahub_command ("(log_file c:/tmp/datahub.log")
```

**Example**

This example creates a new point in the DataHub, and sets its value to 5.

```
if (undefined_p($default:MyNewPoint))
{
    datahub_command ("(cset default:MyNewPoint 5)", 1);
}
```
**datahub_domaininfo**

*datahub_domaininfo* — gives information about data domains.

**Synopsis**

```plaintext
datahub_domaininfo(pattern?)
```

**Arguments**

*pattern*

A character string which specifies a search pattern

**Returns**

An array of instances of the `DH_Domain` class.

**Description**

This function provides information about all data domains whose names match the *pattern*, or all domains if the *pattern* is left blank. The *pattern* can contain the following special characters:

- `*` matches any number of characters, including zero.
- `[c]` matches a single character which is a member of the set contained within the square brackets.
- `[^c]` matches any single character which is not a member of the set contained within the square brackets.
- `?` matches a single character.
- `{xx,yy}` matches either of the simple strings contained within the braces.
- `\c` (a backslash followed by a character) - matches that character.

**Example**

```plaintext
-> datahub_domaininfo();
[[DH_Domain (auto_created . 1) (n_assemblies . 0) (n_attributes . 0)
  (n_bridges . 0) (n_points . 8) (name . "DataSim")]
{DH_Domain (auto_created . 0) (n_assemblies . 0) (n_attributes . 0)
  (n_bridges . 0) (n_points . 6) (name . "MySource")}
{DH_Domain (auto_created . 0) (n_assemblies . 0) (n_attributes . 0)
  (n_bridges . 0) (n_points . 3) (name . "default")}]
```

```plaintext
-> datahub_domaininfo("*Sim");
[[DH_Domain (auto_created . 1) (n_assemblies . 0) (n_attributes . 0)
  (n_bridges . 0) (n_points . 8) (name . "DataSim")]
{DH_Domain (auto_created . 0) (n_assemblies . 0) (n_attributes . 0)
  (n_bridges . 0) (n_points . 6) (name . "MySource")}
```
(n_bridges . 0) (n_points . 8) (name . "DataSim")}
**datahub_domains**

`datahub_domains` — creates a list of all domains.

**Synopsis**

```
datahub_domains()
```

**Arguments**

none

**Returns**

A list of domain names.

**Description**

This function creates a list of all domains currently in the DataHub, except the default domain.

**Example**

```
--> datahub_domains();
(DataSim PlantA PlantB)
```
**datahub_log**

`datahub_log` — writes a string to the Event Log window.

**Synopsis**

```plaintext
datahub_log (trace_level, message)
```

**Arguments**

- `trace_level`
  - One of: DHTL_DEBUG, DHTL_INFO, DHTL_WARNING, DHTL_ERROR.
- `message`
  - Any string.

**Returns**

- `t` on success, otherwise an error message.

**Description**

This function will write a string to the DataHub Event Log window. The `trace_level` parameter corresponds to the level of **Severity** in the Event Log, and the relevant filtering will apply. The **Facility** is always **General**. Messages written with this function will be written to disk if the Event Log is being saved.
**datahub_points**

*datahub_points* — shows the points in a data domain.

**Synopsis**

```ruby
datahub_points (parent, top_level_only? = nil, pattern?)
```

**Arguments**

- **parent**
  - A string containing the name of the parent domain or point name whose child points you wish to list.
- **top_level_only**
  - If `t`, returns only the direct children of the parent. Otherwise, it returns all descendants of the parent.
- **pattern**
  - Returns only those points whose full name (the entire point path without the domain: prefix) matches the pattern.

**Returns**

An array of instances of the `DH_Item` class.

**Description**

This function provides an array containing the DataHub points in a given domain. The `pattern` can contain the following special characters:

- `*` matches any number of characters, including zero.
- `[c]` matches a single character which is a member of the set contained within the square brackets.
- `[^c]` matches any single character which is not a member of the set contained within the square brackets.
- `?` matches a single character.
- `{xx,yy}` matches either of the simple strings contained within the braces.
- `\c` (a backslash followed by a character) - matches that character.

**Example**

```ruby
-> datahub_points("MySource", t);
[(DH_Item (canonical_type . 0) (conf . 0) (domain . "MySource")
  (flags . 307) (n_aliases . 0) (n_attributes . 0) (n_pending . 0))
```
(n_properties . 0) (n_registered . 1) (n_subassemblies . 1)  
(name . "A_Branch") (opcaccessrights . 3) (propid . 0)  
(quality . 0) (security . 0) (timestamp . 0) (value)}

-> datahub_points("MySource", nil, "#em2");
[DH_Item (canonical_type . 5) (conf . 0) (domain . "MySource")  
(flags . 579) (n_aliases . 0) (n_attributes . 0) (n_pending . 0)  
(n_properties . 0) (n_registered . 1) (n_subassemblies . 0)  
(name . "A_Branch.Ramp.X_Branch.Item2") (opcaccessrights . 3)  
(propid . 0) (quality . 192) (security . 0)  
(timestamp . 40200.75880280092) (value . 0)]}
datahub_read

datahub_read — creates a list of all points for a domain.

Synopsis

datahub_read (pointnames ...)

Arguments

pointnames

A DataHub point name as a string, or a list of such DataHub point names as strings. If the first argument is a list, then it is used for pointnames, and all following arguments are ignored. If not, multiple point names can be passed as individual arguments.

Returns

An array in the same order as the specified point names.

Description

This function reads the point identified by each string in pointnames, and returns a DH_Item structure for each point, or nil if the point does not exist.
**datahub_write**

```
datahub_write — assigns a value to a DataHub point.
```

**Synopsis**

```
datahub_write (pointname, value [], force, quality, timestamp)
```

**Arguments**

- **pointname**
  - The name of the point, as a string.

- **value**
  - The value to be assigned to the point—a number, string, or array.

- **force**
  - Causes the write to occur even if the value and quality are the same as the value currently in the DataHub. This is useful if you are changing only the time stamp of the point. Enter `t` to force, or `nil` to not force.

- **quality**
  - The quality of a point, as a number. You can use the constants `OPC_QUALITY_*` found in `GetQualityName` by requiring that file like this: `require("Quality.g");`.

- **timestamp**
  - The time stamp in Windows time. Windows time is a floating point number that can be obtained in Gamma by requiring `Time.g` like this: `require("Time.g");`, then calling the functions `GetCurrentWindowsTime`, `UnixTimeToWindowsTime`, or `PointGetWindowsTime`.

**Returns**

- `t` on success, otherwise an error message.

**Description**

This function is the same as the Gamma `force` function, but the `pointname` is a string, so the symbol never needs to be created in the Gamma engine. This can be used to reduce Gamma engine load when using a large number of points.
**edit_file**

edit_file — opens a file in the Script Editor.

**Synopsis**

```plaintext
default
edit_file (filename)
```

**Arguments**

`filename`

The name of the file you wish to edit, as a string.

**Returns**

t: on success, otherwise an error message.

**Description**

This function opens the Script Editor and loads a requested file. If the requested file cannot be found, it creates a new file with the requested name.
get_point_queue_count

get_point_queue_count — counts the number of DataHub points queued for Gamma.

Synopsis

get_point_queue_count ()

Arguments

none

Returns

On success, the number of points currently queued, as an integer. Otherwise an error message.

Description

This function queries the DataHub for the total number of point changes currently queued to be passed to Gamma. Please refer to _point_queue_depth for a description of the queue. See also get_point_queue_depth.
get_point_queue_depth

get_point_queue_depth — gets the depth of the DataHub's per-point queue for Gamma.

Synopsis

get_point_queue_depth ()

Arguments

none

Returns

On success, the depth of the queue, as an integer. Otherwise an error message.

Description

This function queries the depth of the queue for DataHub changes that get passed to Gamma. Please refer to set_point_queue_depth for a description of the queue. See also get_point_queue_count.
get_tray_menu

get_tray_menu — returns a pointer to the tray menu.

Synopsis

get_tray_menu ()

Arguments

none

Returns

The ID number for the tray menu.

Description

This function returns a pointer to the tray menu.
**on_change**

on_change — evaluates an expression when a variable changes value or quality.

**Synopsis**

```plaintext
on_change (symbol, s_exp)
```

**Arguments**

- `symbol`
  - Any Gamma symbol.
- `s_exp`
  - Any Gamma expression, usually a function or method call.

**Returns**

A list of the following items:

```plaintext
({class_name} fn_name fn_args...)
```

Where `fn_name` and `fn_args` correspond to the `s_exp` parameter.

**Description**

This function causes an expression (`s_exp`) to be evaluated when a variable (`symbol`) changes value or quality. It can be undone with a call to `remove_change`.

**Example**

```plaintext
method AccessData.constructor ()
{
    on_change($DataSim:Sine,
               `(@self).print_point($DataSim:Sine));
    after(3, `destroy(@self));
}
```
**remove_change**

`remove_change` — removes an `on_change` function.

**Synopsis**

```plaintext
remove_change (symbol, s_exp)
```

**Arguments**

- **symbol**
  - Any Gamma symbol.
- **s_exp**
  - Any Gamma expression, usually a function or method call.

**Returns**

A list of the following items:

```plaintext
(class fn_name fn_args ...)
```

Where `fn_name` and `fn_args` correspond to the `s_exp` parameter.

**Description**

This function reverses a call to `on_change`, ending the evaluation of the `s_exp` whenever the `symbol` changes.

**Example**

```plaintext
method AccessData.destructor ()
{
    remove_change(#$DataSim:Sine,
        `(self).print_point($DataSim:Sine));
}
```
**remove_menu_action**

remove_menu_action — removes a menu action.

**Synopsis**

```latex
remove_menu_action (menu_id, s_exp)
```

**Arguments**

- **menu_id**
  A unique ID number for this item.
- **s_exp**
  A Gamma expression that is the action to be removed.

**Returns**

A list:

```
(menu_id  s_exp)
```

Where the `s_exp` is in Lisp format.

**Description**

This function removes a menu action, usually one that was added by `add_menu_action`.

**Example**

This example is part of the `WindowsExample.g` example program:

```gamma
method MonitorWindow.destructor ()
{
...
  try
  {
    with x in .menu_actions do
    remove_menu_action (car(x), cdr(x));
  }
  catch
  {
    princ ("Error: ", _last_error_, "\n");
    print_stack();
  }
...
```
**set_point_flush_flags**

*set_point_flush_flags* — determines which data types are not buffered.

**Synopsis**

```
set_point_flush_flags (flags)
```

**Arguments**

*flags*

Any combination of:

- VT_EMPTY = 0
- VT_I2 = 2
- VT_I4 = 3
- VT_R4 = 4
- VT_R8 = 5
- VT_CY = 6
- VT_DATE = 7
- VT_BSTR = 8
- VT_BOOL = 11
- VT_I1 = 16
- VT_UI1 = 17
- VT_UI2 = 18
- VT_UI4 = 19
- VT_I8 = 20
- VT_UI8 = 21
- VT_INT = 22
- VT_UINT = 23

*item.conf*

A number from 0 to 100 indicating a confidence factor.

*item.flags*

Any bitwise combination of the following numeric set of flags:

- 0x01 Turn on un-buffered checking. Always include this with any other of these flags
- 0x02  Flush on boolean data. If a boolean value changes, transmit all
queued values.

0x04  Flush on integer data. If a boolean value changes, transmit all queued values.

0x08  Flush on floating point data. If a floating point value changes, transmit all queued values.

0x10  Flush on string data. If a string value changes, transmit all queued values.

**Returns**

t on success, otherwise an error message.

**Description**

This function sets which data types will cause the point buffer to immediately be transmitted to the Gamma engine. If *flags* is 0, then the transmission is always buffered. For example, to flush all queued data whenever a boolean or string value changes, make this call:

```
set_point_flush_flags(0x01 | 0x02 | 0x10);
```

To return the Gamma engine to fully buffered transmission, use:

```
set_point_flush_flags(0);
```
set_point_queue_depth

set_point_queue_depth — sets the depth of the DataHub's per-point queue for Gamma.

Synopsis

```
set_point_queue_depth (N)
```

Arguments

N

An integer defining the depth of the queue.

Returns

t on success, otherwise an error message.

Description

The DataHub has a special queueing mechanism for point changes that get sent to Gamma. If the DataHub receives incoming point changes faster than Gamma can process them, it puts the new values into a queue for that point. When Gamma is ready to process the next value of that point, it receives the earliest value from that queue, then the next value, and so on, in time-sequential order.

This function allows you to set the depth of this queue, in other words, the number of values per point that the DataHub will store in the queue. The default depth is 3. We recommend setting the queue depth as small as possible, since a deep queue will decrease Gamma's response time. If you want to minimize the response time and always use the very latest value, you can set $N$ to 0 or 1, which effectively eliminates the queueing behavior altogether.

See also `get_point_queue_depth` and `get_point_queue_count`. 
show_log

show_log — displays the Script Log.

Synopsis

```
show_log ()
```

Arguments

none

Returns

- on success, otherwise an error message.

Description

This function displays the Script Log if it is not already open.
**symcmp**

symcmp — compares symbols to see if they are equal.

**Synopsis**

```
symcmp (symbol, symbol)
```

**Arguments**

- `symbol`
  - Any valid Gamma symbol.

**Returns**

A negative number if the first `symbol` is ordinally less than the second `symbol` according to the ASCII character set, a positive number if the first `symbol` is greater than the second, and zero if the two symbol strings are exactly equal.

**Description**

This function compares symbols to see if they are equal. Neither of the symbols are evaluated when passed to the function. This function can be used as a comparison function for `sort` and `insert`.

**Example**

This example was done in the Script Log.

```
---> symcmp(a,b);
-1
---> symcmp(a,a);
0
---> symcmp(b,a);
1
---> y = array(#c, #d, #a);
[c d a]
---> y = sort(y, symcmp);
[a c d]
---> insert(y, symcmp, #b);
b
---> y;
[a b c d]
```
Methods and Functions from Application.g
AddCustomMenuItem  

AddCustomMenuItem — a convenience method for creating a menu item.

Synopsis

```
AddCustomMenuItem (label, code)
```

Arguments

- `label`  
The name of the item, as a text string, that will actually appear on the menu.

- `code`  
Any piece of code that should be run when the menu item is selected.

Returns

A list containing two members: the ID number for this action, followed by a Gamma expression that is the action to be taken when a menu item is selected.

Description

This method is a convenience method for adding a menu item. It is a wrapper on AddMenuItem. It creates an item for a menu created by AddCustomSubMenu, allowing any arbitrary code to be attached. This method can be used for modifying the application as it runs. Items to be created by this command must be coded immediately after AddCustomSubMenu, and will appear in sequence on that submenu.

Example

```
.AddCustomSubMenu("My Menu", 3);
.AddCustomMenuItem("My First Item, `princ("First Item Activated\n");
.AddCustomMenuItem("My Second Item, `princ("Second Item Activated\n");
.AddCustomMenuItem("My Third Item, `princ("Third Item Activated\n");
...
```
AddCustomSubMenu

AddCustomSubMenu — a convenience method for adding a menu.

Synopsis

AddCustomSubMenu (label, pos?=_TrayMenuPosition)

Arguments

label
The name of the menu, as a text string, that will actually appear on the menu.

pos
An integer designating the position of this item on the menu. This is an optional argument. If not specified, the value of the _TrayMenu class variable will be used.

Returns

The value of the ._CustomSubmenu class variable, a positive integer.

Description

This method is a convenience method for adding a menu. It is a wrapper on AddSubMenu that also automatically assigns the parent menu and arranges its items sequentially as they coded, using the AddCustomMenuItem method.
AddMenuItem

AddMenuItem — adds a menu item and attaches code to it.

Synopsis

AddMenuItem (parent, pos, label, code)

Arguments

parent
The parent menu for this item, such as that returned by a call to CreateSystemMenu or AddSubMenu.

pos
An integer designating the position of this item on the menu.

label
The name of the item, as a text string, that will actually appear on the menu.

code
Any piece of code that should be run when the menu item is selected.

Returns

A list containing two members: the ID number for this action, followed by a Gamma expression that is the action to be taken when a menu item is selected.

Description

This method is used to populate a submenu of the DataHub’s system tray with a selectable item that runs user-specified code when selected.
AddPermanentMenuItem

AddPermanentMenuItem — should not be used.

Synopsis

AddPermanentMenuItem (parent, pos, label, code)

Description

This method should not be used.
AddStartMenuItem

AddStartMenuItem — should not be used.

Synopsis

AddStartMenuItem (label)

Description

This method is for internal use only.
AddStopMenuItem

AddStopMenuItem — creates a menu item that destroys the running application.

Synopsis

AddStopMenuItem (label)

Arguments

label

The name of the item, as a text string, that will actually appear on the menu.

Returns

A list containing two members: the ID number for this action, followed by a Gamma expression that is the action to be taken when a menu item is selected.

Description

This method lets you create an item on a DataHub system tray menu that will exit the application and remove all event handlers and menu items.
AddSubMenu

AddSubMenu — creates a submenu on a parent menu.

Synopsis

AddSubMenu (parent, pos, label)

Arguments

parent
The parent menu for this submenu, such as that returned by a call to CreateSystemMenu or AddSubMenu.

pos
An integer designating the position of this item on the menu.

label
The name of the submenu, as a text string, that will actually appear.

Returns

The value of the ._CustomSubmenu class variable, a positive integer.

Description

This method adds a submenu to the DataHub's system tray menu. The submenu can then be used for adding menu items or other submenus.
ApplicationMultiple

ApplicationMultiple — allows creation of multiple instances of the class.

Synopsis

ApplicationMultiple (klass)

Arguments

klass

Returns

Description

This function does the opposite of ApplicationSingleton. It allows the creation of multiple instances of the Application class, giving each new instance the same name but not the exact same definition.
**ApplicationSingleton**

ApplicationSingleton — allows creation of only one instance of the class.

**Synopsis**

```
ApplicationSingleton (klass)
```

**Arguments**

```
klass
```

The name of a class.

**Returns**

The instance of the class named by `klass`.

**Description**

This function creates a singleton based on the name of a class, which in most cases is desirable. This is useful because a class is routinely redefined by reloading its corresponding file, The alternative, `ApplicationMultiple` gives instances of the class the same name but not the same definition.
CreateSystemMenu

CreateSystemMenu — adds a submenu to the system tray menu.

Synopsis

CreateSystemMenu ()

Arguments

none

Returns

The value of the .CustomSubmenu class variable, a positive integer.

Description

This method adds a Scripts submenu to the DataHub's system tray menu, which can then be used for adding submenus or menu items for DataHub scripts.
droptimer

droptimer — for internal use only.

Synopsis

droptimer (tid)

Description

This method is for internal use only.
OnChange

OnChange — attaches an event handler to a point change event.

Synopsis

OnChange (sym, fn)

Arguments

sym

Any Gamma symbol.

fn

A function or method call.

Returns

A list containing the sym and fn.

Description

This method is a wrapper for the on_change function. In addition to applying that function, it adds this change to the class's _ChangeFunctions list. For more information, please refer to the section called “Handling Events”.
RemoveAllChanges

RemoveAllChanges — removes event handlers from all point change events.

Synopsis

RemoveAllChanges ()

Arguments

none

Returns

nil on success, otherwise an error.

Description

This method removes all changes from the class's _ChangeFunctions list, and reassigns the _ChangeFunctions variable to nil. For more information, please refer to the section called “Handling Events”.
RemoveAllEventHandlers

RemoveAllEventHandlers — removes all point change events and all timers.

Synopsis

RemoveAllEventHandlers ()

Arguments

none

Returns

nil on success, otherwise an error.

Description

This method removes all change events and all timers by calling RemoveAllChanges and RemoveAllTimers.
RemoveAllMenus

RemoveAllMenus — removes all script menus, submenus, and menu actions.

Synopsis

RemoveAllMenus ()

Arguments

none

Returns

nil on success, or an error.

Description

This method removes all script menus, submenus, menu items, and actions.
RemoveAllTimers

RemoveAllTimers — cancels all timers.

Synopsis

RemoveAllTimers ()

Arguments

none

Returns

nil on success, or an error.

Description

This method cancels all timers and sets the ._TimerIDs list to nil.
**RemoveChange**

RemoveChange — removes an event handler from a point change event.

**Synopsis**

RemoveChange (chfn)

**Arguments**

*chfn*

A change function, as created by `OnChange`.

**Returns**

A list of the following items:

`(class fn_name fn_args ...)`

Where *fn_name* and *fn_args* are the name and arguments of the removed function.

**Description**

This method is a wrapper for the `remove_change` function. In addition to applying that function, it removes this change from the class's `_ChangeFunctions` list. For more information, please refer to the section called "Handling Events".
**RemoveSystemMenu**

*RemoveSystemMenu* — for internal use only.

**Synopsis**

```c
RemoveSystemMenu()
```

**Description**

This method is for internal use only.
RemoveTimer

RemoveTimer — cancels a timer.

Synopsis

```
RemoveTimer(tid)
```

Arguments

`tid`

A timer ID number, as assigned by `TimerAt`, `TimerAfter`, or `TimerEvery`.

Returns

The `_TimerIDs` list with the `tid` removed.

Description

This method cancels the timer corresponding to `tid`, and removes that timer ID number from the `_TimerIDs` list.
TimerAfter

TimerAfter — attaches an event handler to an "after" timer.

Synopsis

TimerAfter (seconds, fn)

Arguments

seconds
A number of seconds

fn
A function or method call.

Returns

An integer that is a timer ID number.

Description

This method sets an after timer that fires after the number of seconds specified, causing the fn function or method to execute. This method also creates a unique, sequential ID number for the timer, appends that number to the class's ._TimerIDs list, and returns that same timer ID number.

Timers created using TimerAt, TimerAfter, and TimerEvery are automatically cancelled when the Application instance is destroyed.
TimerAt

TimerAt — attaches an event handler to an "at" timer.

Synopsis

TimerAt (day, month, year, hour, minute, second, fn)

Arguments

day
Restriction on the day of the month (1-31), or nil for none.

month
Restriction on the month of the year (1-12), or nil for none.

year
Restriction on the year (1994-2026), or nil for none.

hour
Restriction on the hour of the day (0-23), or nil for none.

minute
Restriction on the minute in the hour (0-59), or nil for none.

second
Restriction on the second in the minute (0-59), or nil for none.

fn
A function or method call.

Returns

Description

This method sets an at timer that causes the fn function or method to execute at a specific time, or to occur regularly at certain times of the minute, hour, day, month or year. A restriction on a particular attribute of the time will cause the timer to fire only if that restriction is true.

A restriction may be any number in the legal range of that attribute, or a list of numbers in that range. Illegal values for the time will be normalized. For example, a time specified as July 0, 2008 00:00:00 will be treated as June 30, 2008 00:00:00. If nil is specified for any attribute of the time, this implies no restriction and the timer will fire cyclically at every legal value for that attribute.

This method also creates a unique, sequential ID number for the timer, appends that number to the class's ._TimerIDs list, and returns that same timer ID number.
Timers created using `TimerAt`, `TimerAfter`, and `TimerEvery` are automatically cancelled when the `Application` instance is destroyed.
**TimerEvery**

TimerEvery — attaches an event handler to an "every" timer.

**Synopsis**

```
TimerEvery (seconds, fn)
```

**Arguments**

*seconds*

A number of seconds.

*fn*

A function or method call.

**Returns**

An integer that is a timer ID number.

**Description**

This method sets an *every* timer that that fires periodically every number of *seconds*, causing the *fn* function or method to execute. This method also creates a unique, sequential ID number for the timer, appends that number to the class's .*_TimerIDs* list, and returns that same timer ID number.

Timers created using TimerAt, TimerAfter, and TimerEvery are automatically cancelled when the Application instance is destroyed.
Time Conversion Functions from Time.g

The DataHub works in two different time standards:

- Windows DATE (VT_DATE) is the number of days, as an 8-byte float since December 30, 1899. The fractional part is unsigned, so 6 a.m. December 29, 1899 would be represented as -1.25.

- Unix time is the number of seconds since 12 a.m. January 1, 1970. It is a signed 4-byte integer. Negative times have no meaning.

In these functions, Unix time uses the traditional Unix time plus a number of nanoseconds. Since this cannot be expressed as a single 4-byte integer, we express it as an 8-byte floating point number where the fractional part adds accuracy. This accuracy is ideally measured down to the nanosecond, but in practice relies on the accuracy of the operating system clock. In Windows, this means milliseconds.

Unix time is important because it is the time format used in the Gamma functions `time`, `mktime`, `localtime` and `date`. It is also the time format used when explicitly specifying a time stamp on a point in Unix.

Windows time is the internal time representation used by DataHub points.

The functions listed here are created in the script `Time.g`, and are available to any DataHub script that requires or includes the `Time.g` file, using one of these statements:

- `require ("Time");`
- `include ("Time");`
GetCurrentWindowsTime

GetCurrentWindowsTime — returns the current clock time in Windows time format.

Synopsis

```c
GetCurrentWindowsTime ()
```

Arguments

none

Returns

The current clock time, in Windows time format.
**PointGetUnixTime**

PointGetUnixTime — gets the Unix time stamp from a point.

**Synopsis**

```c
PointGetUnixTime (point)
```

**Arguments**

`point`

The fully-qualified symbolic point name of a DataHub point.

**Returns**

The Unix time stamp for the point.
PointGetWindowsTime

PointGetWindowsTime — gets the Windows time stamp from a point.

Synopsis

PointGetWindowsTime (point)

Arguments

point
The fully-qualified symbolic point name of a DataHub point.

Returns

The Windows time stamp.
**PointMetadata**

PointMetadata — queries a point for its metadata structure.

**Synopsis**

```
PointMetadata (point)
```

**Arguments**

`point`

The fully-qualified symbolic point name of a DataHub point.

**Returns**

Either a `DH_Item` structure, or `nil`.

**Description**

This function queries a point for its metadata structure, which is a `DH_Item` containing the following fields:

- `item.canonical_type`
  A number representing the canonical type of the point. See the possible values of `VAR_TYPE` in Windows:

  - `DH_ITEM_READABLE 0x0001`
  - `DH_ITEM_WRITEABLE 0x0002`
  - `DH_ITEM_LOCKED 0x0004`
  - `DH_ITEM_PROPERTY 0x0008`
  - `DH_ITEM_SUBASSEMBLY 0x0010`
  - `DH_ITEM.Assembly 0x0020`
  - `DH_ITEM_ATTRIBUTE 0x0040`
  - `DH_ITEM_TYPE 0x0080`
  - `DH_ITEM_PRIVATE 0x0100`
  - `DH_ITEM_PRI_SYMBOLIC`
item.opcaccessrights
Any bitwise combination of 1 for READABLE and 2 for WRITABLE. The same as the first two bits of item.flags.

item.quality
The OPC quality value. You can get the name associated with the value by using the GetQualityName function in the Quality.g file. Put the statement:

```javascript
require ("Quality");
```
at the beginning of your script to gain access to this function.

item.scan_max
Maximum scan rate on this point. Not in use.

item.security
The security level on this point.

item.timeoffset
Offset in seconds from local clock time for the originator of this point value. Not in use.

item.timestamp
The Windows time stamp of this point.

The point metadata is likely to change in the future, though the actual fields available will remain the same. It would be good practice to wrap access to the metadata within wrapper functions.
**UnixLocalToUTC**

UnixLocalToUTC — converts a local Unix time into UTC.

**Synopsis**

UnixLocalToUTC (secs, offsetsecs)

**Arguments**

*secs*

Unix time, in seconds.

*offsetsecs*

The offset between local time and UTC, in seconds.

**Returns**

The UTC time.
UnixTimeToWindowsTime

UnixTimeToWindowsTime — converts from Unix time to Windows time.

Synopsis

UnixTimeToWindowsTime (secs)

Arguments

secs

Unix time, in seconds.

Returns

A single number representing Windows time.
UnixUTCToLocal

UnixUTCToLocal — converts a UTC Unix time into local time.

Synopsis

UnixUTCToLocal (secs, offsetsecs)

Arguments

secs

Unix time, in seconds.

offsetsecs

The offset between local time and UTC, in seconds.

Returns

A local time.
WindowsLocalToUTC

WindowsLocalToUTC — converts a local Windows time into UTC.

Synopsis

WindowsLocalToUTC (wdate, offsetsecs)

Arguments

wdate
  The date, in Windows time.

offsetsecs
  The offset between local time and UTC, in seconds.

Returns

A UTC time.
**WindowsTimeToUnixTime**

WindowsTimeToUnixTime — converts from Windows time to Unix time.

**Synopsis**

```plaintext
WindowsTimeToUnixTime (wdate)
```

**Arguments**

- `wdate`  
  The date, in Windows time.

**Returns**

The date, in Unix time.

**Description**

This function converts from Windows time to Unix time. If the `wdate` is prior to January 1, 1970, the function returns `nil`. If `wdate` is after the end of the Unix epoch (03:14:08 UTC on January 19, 2038) then the returned value will be greater than the acceptable input range for Unix date-based functions.
WindowsUTCToLocal

WindowsUTCToLocal — converts a UTC Windows time into local time.

Synopsis

```c
WindowsUTCToLocal (wdate, offsetsecs)
```

Arguments

- `wdate`:
  The date, in Windows time.
- `offsetsecs`:
  The offset between local time and UTC, in seconds.

Returns

A local Windows time.
Regular Expression Methods from RegexSupport.g

The RegexSupport.g file provides support for the Perl-Compatible Regular Expression (PCRE) library (http://www.pcre.org/) and includes the PCRS substitution extensions to PCRE.

You must require the Regex support methods before you can use them in a script. At the top of your script, include the following line:

```perl
require (RegexSupport);
```

The RegexSupport.g file provides three classes:

• **Regex**, a compiled regular expression pattern. All but one of the methods in this reference are for this class.

• **pcrs_job**, a compiled substitution expression. There is one method for this class, `pcrs_job.Exec`.

• **pcre_extra**, which contains hints for the regular expression Exec function to speed up execution.

Please see the **Match** and **Subst** entries for examples.
Compile

Compile — compiles a regular expression to a form that is more efficient to evaluate.

Synopsis

```csharp
static Regex.Compile(pattern, options)
```

Arguments

- **pattern**
  - A string specifying the regular expression pattern.

- **options**
  - A bitwise OR of zero or more of the following:
    - `PCRE_ANCHORED`
    - `PCRE_BSR_ANYCRLF`
    - `PCRE_BSR_UNICODE`
    - `PCRE_CASELESS`
    - `PCRE_DOLLAR_ENDONLY`
    - `PCRE_DOTALL`
    - `PCRE_DUPNAMES`
    - `PCRE_EXTENDED`
    - `PCRE_EXTRA`
    - `PCRE_FIRSTLINE`
    - `PCRE_JAVASCRIPT_COMPAT`
    - `PCRE_MULTILINE`
    - `PCRE_NEWLINE.CR`
    - `PCRE_NEWLINE.LF`
    - `PCRE_NEWLINE.CRLF`
    - `PCRE_NEWLINE.ANYCRLF`
    - `PCRE_NEWLINE.ANY`
    - `PCRE_NO_AUTO_CAPTURE`
    - `PCRE_UNGREEDY`
    - `PCRE_UTF8`
    - `PCRE_NO_UTF8_CHECK`

Returns

On success, an instance of `Regex`. On failure, an error list with three elements:

- `car(errlist)` = a numeric return code from the failing function
- `cadr(errlist)` = a character position in the pattern where the error occurred, or 0.
- `caddr(errlist)` = an error message, as a human-readable string
See Also

Match
CompileSubst

CompileSubst — compiles a substitution pattern.

Synopsis

```c
static Regex.CompileSubst(pattern)
```

Arguments

`pattern`

A pattern of the form `s/pattern/substitution/flags`.

Returns

On success, an instance of `pcrs_jobs`. On failure, an error list with three elements:

- `car(errlist)` = a numeric return code from the failing function
- `cadr(errlist)` = a character position in the pattern where the error occurred, or 0.
- `caddr(errlist)` = an error message, as a human-readable string

Description

This method compiles a pattern of the form `s/pattern/substitution/flags`. 
CompileSubstEx

CompileSubstEx — an alternate interface to CompileSubst.

Synopsis

```
static Regex.CompileSubstEx(pattern, substitution, flags?=0)
```

Arguments

- `pattern`
  Normally, a string, but optionally a buffer. If a buffer, it may contain NUL characters.
- `substitution`
  Normally, a string, but optionally a buffer. If a buffer, it may contain NUL characters.
- `flags`
  A string.

Returns

On success, an instance of `pcrs_job`. On failure, an error list with three elements:

- `car` (errlist) = a numeric return code from the failing function
- `cadr` (errlist) = a character position in the pattern where the error occurred, or 0.
- `caddr` (errlist) = an error message, as a human-readable string
Config

Config — queries the PCRE library for compiled-in options.

Synopsis

```c
static Regex.Config(what)
```

Arguments

`what`

The option to query.

Returns

A number.

Description

See the documentation for `pcre_config` for details.
Exec

Exec — applies a regular expression match to a subject string.

Synopsis

Regex.Exec(extra, subject, length, startoffset, options, substrings)

Arguments

extra
An instance of pcre_extra returned from Regex.Study, or nil.

subject
The subject string on which to apply the pattern. This may be either a string or a buffer. If the subject is a buffer it may contain NUL characters.

length
The length of the subject, in bytes. If length is -1, the length of the subject string is computed automatically.

startoffset
A byte offset into the subject specifying at which point to start.

options
Option flags - a bitwise OR of zero or more of these flags:

PCRE_ANCHORED
PCRE_NEWLINE_CR
PCRE_NEWLINE_LF
PCRE_NEWLINE_CRLF
PCRE_NEWLINE_ANYCRLF
PCRE_NEWLINE_ANY
PCRE_NOTBOL
PCRE_NOTEOL
PCRE_NOTEMPTY
PCRE_NOTEMPTY_ATSTART
PCRE_NO_START_OPTIMIZE
PCRE_NO_UTF8_CHECK
PCRE_PARTIAL_SOFT
PCRE_PARTIAL_HARD

substrings
An array into which the match substrings are copied. The substrings array cannot be nil. The match substrings will be written into the substrings array starting at position 0. If the array is not large enough to hold all of the substrings it will be automatically expanded.
Each element of the substrings array is itself an array of 3 elements:

[0] - the zero-based starting character of the match  
[1] - the character position immediately after the match  
[2] - the string that matched.  
This will be returned as a buffer if the subject is a buffer, otherwise it will be a string.  
The first element in the substrings array always contains the complete matching string within the subject. Each element after the first corresponds to a positional substring specified within the pattern.

Returns

On success, an array of match specifications as described in the substrings parameter (above). On failure, an error list with three elements:

• car (errlist) = a numeric return code from the failing function  
• cadr (errlist) = a character position in the pattern where the error occurred, or 0.  
• caddr (errlist) = an error message, as a human-readable string

See Also

Match
**pcrs_job.Exec**

*pcrs_job.Exec* — applies a compiled pattern substitution to a subject string.

**Synopsis**

```
pcrs_job.Exec(subject, length?=-1)
```

**Arguments**

- **subject**
  - A subject string for substitution.
- **length**
  - The length of the subject string. Entering -1 (the default) for this parameter will cause the length of the subject string to be automatically computed.

**Returns**

On success, the new string with the substitutions applied. On failure, an error list with three elements:

- **car (errlist)** = a numeric return code from the failing function
- **cadr (errlist)** = a character position in the pattern where the error occurred, or 0.
- **caddr (errlist)** = an error message, as a human-readable string

**Description**

This method of the *pcrs_job* class applies a previously compiled pattern substitution to a subject string.
**GetStringNumber**

GetStringNumber — retrieves the index number of named substrings.

**Synopsis**

```
Regex.GetStringNumber(name)
```

**Arguments**

- `name` 
  The name of a substring.

**Returns**

On success, a number greater than 1. On failure, a numeric error code less than 0.

- `car(errlist)` = a numeric return code from the failing function
- `cadr(errlist)` = a character position in the pattern where the error occurred, or 0.
- `caddr(errlist)` = an error message, as a human-readable string

**Description**

If a pattern contains named substring matches, this method will retrieve the index number (starting at 1) of the substring corresponding to this name.
Match

Match — a convenience function that combines calls to Compile, Study, and Exec.

Synopsis

```csharp
static Regex.Match(pattern, subject, options?=0,
        length?=-1, startoffset?=0)
```

Arguments

- **pattern**
  A string specifying the regular expression pattern.

- **subject**
  The subject string on which to apply the pattern. This may be either a string or a buffer. If the subject is a buffer it may contain NUL characters.

- **options**
  Option flags - a bitwise OR of zero or more of these flags:

  - PCRE_ANCHORED
  - PCRE_BSR_ANYCRLF
  - PCRE_BSR_UNICODE
  - PCRE_CASELESS
  - PCRE_DOLLAR_ENDONLY
  - PCRE_DOTALL
  - PCRE_DUPNAMES
  - PCRE_EXTENDED
  - PCRE_EXTRA
  - PCRE_FIRSTLINE
  - PCRE_JAVASCRIPT_COMPAT
  - PCRE_MULTILINE
  - PCRE_NEWLINE_ANY
  - PCRE_NEWLINE_ANYCRLF
  - PCRE_NEWLINE_CR
  - PCRE_NEWLINE_CRLF
  - PCRE_NEWLINE_LF
  - PCRE_NOTBOL
  - PCRE_NOTEMPTY
  - PCRE_NOTEMPTY_ATSTART
  - PCRE_NOTEOL
  - PCRE_NO_AUTO_CAPTURE
  - PCRE_NO_START_OPTIMIZE
  - PCRE_NO_UTF8_CHECK
  - PCRE_PARTIAL_HARD
Regular Expression Methods from RegexSupport.g

PCRE_PARTIAL_SOFT
PCRE_UNGREEDY
PCRE_UTF8

length
The length of the subject, in bytes. If length is −1 (the default), the length of the subject string is computed automatically.

startoffset
A byte offset into the subject specifying at which point to start.

Returns
On success, an array of match specifications equivalent to the substrings argument to Regex.Exec. On failure, an error list with three elements:

• car(errlist) = a numeric return code from the failing function
• cadr(errlist) = a character position in the pattern where the error occurred, or 0.
• caddr(errlist) = an error message, as a human-readable string

Examples
Find and return all matches of hello, case sensitive:

Gamma> Regex.Match("(hello)", "I say Hello and you say Goodbye");
(-1 0 "No match found")

Find and return all matches of hello, case insensitive:

Gamma> Regex.Match("(hello)", "I say Hello and you say Goodbye", PCRE_CASELESS);
[[0 31 #{I say Hello and you say Goodbye}] [6 11 #{Hello}]]

See Also

Compile, Study, Exec, and Subst
Study

Study — helpful for speeding up repeated applications of a regular expression.

Synopsis

```
Regex.Study(options?=0)
```

Arguments

```
options
```

Must be zero in this implementation.

Returns

On success, an instance of `pcre_extra`. On failure, an error list with three elements:

- `car(errlist)` = a numeric return code from the failing function
- `cadr(errlist)` = a character position in the pattern where the error occurred, or 0.
- `caddr(errlist)` = an error message, as a human-readable string

Description

This method evaluates a regular expression to produce hints that can speed up the application of the regular expression in future. This method does not need to be called, but is useful to speed up repeated application of the same regular expression.

See Also

`Match`
Subst

Subst — applies a substitution pattern to a string of a given length.

Synopsis

```c
static Regex.Subst(pattern, subject, length?=-1)
```

Arguments

- **pattern**
  A substitution pattern, in the form `s/pattern/substitution/flags`.

- **subject**
  The subject string on which the substitution will be made.

- **length**
  The length of the subject string. Entering `-1` (the default) for this parameter will cause the length of the subject string to be automatically computed.

Returns

On success, returns the new string with the substitutions applied. On failure, an error list with three elements:

- **car** (errlist) = a numeric return code from the failing function
- **cadr** (errlist) = a character position in the pattern where the error occurred, or 0.
- **caddr** (errlist) = an error message, as a human-readable string

Examples

String replacement:

```c
Gamma> x = "This is a test of my test system"
"This is a test of my test system"
Gamma> Regex.Subst("s/test/build/g", x)
"This is a build of my build system"
```

To surround all words with parentheses:

```c
Gamma> Regex.Subst ("s/(\w+)/\(($1\))/g", "This is a test!");
"(This) (is) (a) (test)!"
```

See Also

- **Match**
Quality Name Function from Quality.g
GetQualityName

GetQualityName — converts the quality value of a point to a text string.

Synopsis

GetQualityName (quality)

Arguments

quality
A value that indicates the quality of the point, as returned by the function PointMetaData.

Returns

A text string corresponding to the quality.

Description

This function converts the quality value of a point to a text string. The function is created in the script Quality.g, and is available to any DataHub script that requires or includes the Quality.g file, using one of these statements:

• require ("Quality");
• include ("Quality");

DataHub point information includes a data quality indication that is consistent with the OPC specification. Even if you are not working with OPC data, you may still wish to use the quality in your custom applications and DataHub scripts.

The possible quality strings are:

<table>
<thead>
<tr>
<th>Value</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC_QUALITY_BAD</td>
<td>Bad</td>
</tr>
<tr>
<td>OPC_QUALITY_COMM_FAILURE</td>
<td>Comm Failure</td>
</tr>
<tr>
<td>OPC_QUALITY_CONFIG_ERROR</td>
<td>Config Error</td>
</tr>
<tr>
<td>OPC_QUALITY_DEVICE_FAILURE</td>
<td>Device Failure</td>
</tr>
<tr>
<td>OPC_QUALITY_EGU_EXCEEDED</td>
<td>EGU Exceeded</td>
</tr>
<tr>
<td>OPC_QUALITY_GOOD</td>
<td>Good</td>
</tr>
<tr>
<td>OPC_QUALITY_LAST_KNOWN</td>
<td>Last Known</td>
</tr>
<tr>
<td>OPC_QUALITY_LAST_USABLE</td>
<td>Last Usable</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>OPC качества LOCAL OVERRIDE</td>
<td>Local Override</td>
</tr>
<tr>
<td>OPC качества MASK</td>
<td>Mask</td>
</tr>
<tr>
<td>OPC качества NOT CONNECTED</td>
<td>Not Connected</td>
</tr>
<tr>
<td>OPC качества OUT OF SERVICE</td>
<td>Out Of Service</td>
</tr>
<tr>
<td>OPC качества SENSOR CAL</td>
<td>Sensor Cal</td>
</tr>
<tr>
<td>OPC качества SENSOR FAILURE</td>
<td>Sensor Failure</td>
</tr>
<tr>
<td>OPC качества SUB NORMAL</td>
<td>Sub Normal</td>
</tr>
<tr>
<td>OPC качества UNCERTAIN</td>
<td>Uncertain</td>
</tr>
</tbody>
</table>
Classes from HistorianSupport.g

The functionality of the DataHub Historian features is made available through a Historian class provided in the HistorianSupport.g script shipped with the DataHub.

You must require the Historian support methods before you can use them in a script. At the top of your script, include the following line:

```plaintext
require (HistorianSupport);
```

The HistorianSupport.g file provides three classes: Historian, HistoryBuffer, and HistoryValue. The HistTest.g script provides an example of how these classes are used to query the DataHub Historian.
Historian

Historian — encapsulates the Historian facility in the DataHub.

Synopsis

class Historian
{
}

Instance Variables

None.

Methods

allowAutoHistoryAdds(state, defaultgroup)
Configures automatic history addition. The state should be nil to turn off the function and t to turn it on. The defaultgroup is a string naming the History Group to which automatic histories are added. This method corresponds to this setting in the Historian option of the Properties window:

addGroup(label, basedir, cachesize)
Creates a new History Group with the provided label, basedir (base directory) and cachesize (cache size). This corresponds approximately to pressing the Add ... button in the Historian option of the Properties window.

removeGroup(label)
Removes the History Group with the given label.

setFileTimes(labelpattern, days, hours, minutes)
Assigns the file times for all History Groups whose labels match the labelpattern. The pattern uses the pattern matching rules documented in the shell_match function. This method corresponds to this setting in the Properties window Historian configuration:

setRentionTimes(labelpattern, days, hours, minutes)
Assigns the data retention times for all History Groups whose labels match the labelpattern. The pattern uses the pattern matching rules documented in the shell_match function. This method corresponds to this setting in the Properties window Historian configuration:
setFlushTimes(labelpattern, days, hours, minutes), ()
Assigns the file flush times for all History Groups whose labels match the labelpattern. The pattern uses the pattern matching rules documented in the shell_match function. This method corresponds to this setting in the Properties window Historian configuration:

![Flush to disk settings](image)

setDeadband(labelpattern, flags, absolute, percent, maxsecs, maxcount)
Assigns the deadband settings for all History Groups whose labels match the labelpattern. The pattern uses the pattern matching rules documented in the shell_match function. This method corresponds to this section of the Properties window Historian configuration:

![Deadband settings](image)

The flags argument is 0 to turn off all deadbands, or any combination of:
- 0x0008 - Turn on the percent deadband
- 0x0020 - Turn on the absolute deadband
- 0x0040 - Turn on the maximum time limit
- 0x0080 - Turn on the maximum skip count

addPoint(pointname, grouplabel?=nil)
Adds a point to the named History Group. If grouplabel is nil or not specified then the point will be added to the automatic group. If the automatic group is not set and the grouplabel is not specified, this method will fail silently.
The pointname is a string specifying the fully qualified point name. That is, the point name must contain the data domain and the full path of the point, for example, DataPid:PID1.Mv

removePoint(pointpattern, grouppattern?="")
Removes all points matching the pointpattern from all groups matching the grouppattern.

saveConfig()
Causes the Historian to save its configuration to disk.

startConfiguring()
Tells the Historian to enter configuring mode. In this mode, changes to group configurations are accepted but not applied. This allows a number of configuration commands to be sent without affecting the current state of the Historian. This method must eventually be followed by a call to endConfiguring for the configuration changes to be applied.
Classes from HistorianSupport.g 189

endConfiguring()
Tells the Historian to exit configuring mode, and to apply any changes to the configuration that have been made since the call to startConfiguring.

queryRawData(pointname, start_unixtime, duration_secs)
Queries the history for a given point, starting at the specified time (start_unixtime) and running for the specified number of seconds (duration_secs). The pointname must be a string containing the fully qualified name of the point. The start_unixtime must be specified as number of seconds since midnight January 1, 1970 UTC. Both the start_unixtime and duration_secs are floating point values and can contain a fractional component.
The return value from this function is either nil or an instance of HistoryBuffer containing the values that fall within the specified range.
The data returned from this method is the exact data set as stored on disk without modification.

queryStatistic(pointname, start_unixtime, duration_secs, aggregation_period_secs, stat_name)
Queries a statistical measure of the history for a given point, starting at the specified time (start_unixtime) and running for the specified number of seconds (duration_secs). The pointname must be a string containing the fully qualified name of the point. The start_unixtime must be specified as number of seconds since midnight January 1, 1970 UTC. Both the start_unixtime and duration_secs are floating point values and can contain a fractional component.
The return value from this function is either nil or an instance of HistoryBuffer containing the values that fall within the specified range.
The aggregation_period_secs specifies the number of seconds in each time interval within the overall query on which to produce statistics. For example, a query may request the average value in each 5-second period for a minute. The aggregation period would be 5 seconds, and the duration would be 60 seconds. The result set will contain 12 values, one for each 5-second interval within the duration of the query.
The stat_name is a string that specifies which statistical measure to query. The statistical measures available are:

<table>
<thead>
<tr>
<th>Statistic Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Retrieve the totalized value (time integral) of the data over the sample interval.</td>
</tr>
<tr>
<td>Average</td>
<td>Retrieve the average data over the sample interval.</td>
</tr>
<tr>
<td>TimeAverage</td>
<td>Retrieve the time weighted average data over the sample interval.</td>
</tr>
<tr>
<td>Count</td>
<td>Retrieve the number of good quality raw values over the sample interval.</td>
</tr>
<tr>
<td>RawCount</td>
<td>Retrieve the number of raw values over the sample interval.</td>
</tr>
<tr>
<td>StDev</td>
<td>Retrieve the standard deviation over the sample interval.</td>
</tr>
<tr>
<td>Statistic Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MinimumActualTime</td>
<td>Retrieve the minimum value in the sample interval and the timestamp</td>
</tr>
<tr>
<td></td>
<td>of the minimum value.</td>
</tr>
<tr>
<td>Minimum</td>
<td>Retrieve the minimum value in the sample interval.</td>
</tr>
<tr>
<td>MaximumActualTime</td>
<td>Retrieve the maximum value in the sample interval and the timestamp</td>
</tr>
<tr>
<td></td>
<td>of the maximum value.</td>
</tr>
<tr>
<td>Maximum</td>
<td>Retrieve the maximum value in the sample interval.</td>
</tr>
<tr>
<td>Start</td>
<td>Retrieve the value at the beginning of the sample interval. The time stamp</td>
</tr>
<tr>
<td></td>
<td>is the time stamp of the beginning of the interval.</td>
</tr>
<tr>
<td>End</td>
<td>Retrieve the value at the end of the sample interval. The time stamp is</td>
</tr>
<tr>
<td></td>
<td>the time stamp of the end of the interval.</td>
</tr>
<tr>
<td>First</td>
<td>Retrieve the first value change within the interval. The time stamp is</td>
</tr>
<tr>
<td></td>
<td>the time stamp of the first value change.</td>
</tr>
<tr>
<td>Last</td>
<td>Retrieve the last value change within the interval. The time stamp is</td>
</tr>
<tr>
<td></td>
<td>the time stamp of the last value change.</td>
</tr>
<tr>
<td>Delta</td>
<td>Retrieve the difference between the first and last value in the sample</td>
</tr>
<tr>
<td></td>
<td>interval.</td>
</tr>
<tr>
<td>RegSlope</td>
<td>Retrieve the slope of the regression line over the sample interval.</td>
</tr>
<tr>
<td>RegConst</td>
<td>Retrieve the starting point of the regression line over the sample</td>
</tr>
<tr>
<td></td>
<td>interval. This is the value of the regression line at the start of the</td>
</tr>
<tr>
<td></td>
<td>interval.</td>
</tr>
<tr>
<td>RegPearsonR</td>
<td>Retrieve Pearson R measure of correlation between Y and Time over the</td>
</tr>
<tr>
<td></td>
<td>sample interval. This is a number between -1.0 and 1.0.</td>
</tr>
<tr>
<td>RegRSquared</td>
<td>Retrieve R-squared measure of regression fitness over the sample interval.</td>
</tr>
<tr>
<td></td>
<td>This is a number between 0 and 1.0.</td>
</tr>
<tr>
<td>Variance</td>
<td>Retrieve the variance over the sample interval.</td>
</tr>
<tr>
<td>Range</td>
<td>Retrieve the difference between the minimum and maximum value over the</td>
</tr>
<tr>
<td></td>
<td>sample interval.</td>
</tr>
<tr>
<td>DurationGood</td>
<td>Retrieve the duration (in seconds) of time in the interval during which the</td>
</tr>
<tr>
<td></td>
<td>data is good.</td>
</tr>
<tr>
<td>DurationBad</td>
<td>Retrieve the duration (in seconds) of time in the interval during which the</td>
</tr>
<tr>
<td></td>
<td>data is bad.</td>
</tr>
<tr>
<td>PercentGood</td>
<td>Retrieve the percent of data (1 equals 100 percent) in the interval which</td>
</tr>
<tr>
<td></td>
<td>has good quality.</td>
</tr>
<tr>
<td>PercentBad</td>
<td>Retrieve the percent of data (1 equals 100 percent) in the interval which</td>
</tr>
<tr>
<td></td>
<td>has bad quality.</td>
</tr>
<tr>
<td>WorstQuality</td>
<td>Retrieve the worst quality of data in the interval.</td>
</tr>
<tr>
<td>Statistic Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ValueSum</td>
<td>Retrieve the sum of all raw values over the sample interval.</td>
</tr>
</tbody>
</table>
HistoryBuffer

HistoryBuffer — a set of HistoryValues returned from a Historian query.

Synopsis

class HistoryBuffer
{
    npoints;
}

Instance Variables

npoints
    The number of values (of type HistoryValue) in this data set.

Methods

Result sets from historical queries can be large. Consequently the access methods for a
result set come in two types. The first type, named getSomething, will return a reference
to the particular data array or item, without making a copy. This reference is only valid un-
til the HistoryBuffer instance is destroyed, which may happen at any time after all ref-
erences to the HistoryBuffer instance go out of scope.

⚠️ If the HistoryValue or array returned from the get method is used after the
HistoryBuffer is destroyed, the DataHub could crash.

Alternately, you can use the copySomething methods to create duplicates of the da-
ta requested. These instances will persist after the HistoryBuffer is destroyed. The
copy methods are more robust, but increase the CPU usage and memory usage of large
queries. In general you should use the copy methods when you are unsure whether the
lifetimes of the HistoryValues will exceed the lifetime of the HistoryBuffer.

copyToArray()
    Copy the internal data representation to an array of HistoryValue instances. This
    method returns the new array. This method is similar to toArray, except that it
    makes a copy of each HistoryValue instance.

copyValue(index)
    Returns a copy of the HistoryValue at the given index within the data set. The in-
    dex is a number starting at 0 and less then npoints. This is the safe version of the
    getValue method.

count()
    Returns the number of values in this data set.
**getValue(index)**

Returns the HistoryValue at the given index in the data set. This is the unsafe version of copyValue. It will return a reference to the HistoryValue within the HistoryBuffer. This reference will only be valid until the HistoryBuffer is destroyed.

**setValue(index, histvalue)**

Modifies the original data in the HistoryBuffer at the given index. This method will result in changes to the original data, and to any HistoryValue instances that have been acquired by reference using getValue or toArray. The histvalue argument is an instance of HistoryValue.

**toArray()**

Returns an array of HistoryValue instances as references to the original data. This is equivalent to creating an array by repeatedly calling getValue. The resulting data is only valid until the HistoryBuffer is destroyed.
HistoryValue

HistoryValue — a single value from the Historian, consisting of value, x-axis and quality.

Synopsis

```java
class HistoryValue {
    xaxis;
    value;
    quality;
}
```

Instance Variables

`xaxis`

The X axis value, generally the time stamp in UNIX time for this value. In some cases, a query can return an Y-vs-X result set instead of a Y-vs-Time result set. Consequently the X axis may not be time in all cases. This is a floating point number that can represent fractional seconds.

`value`

The data value. This is either a calculated value or a raw value, depending on the type of query.

`quality`

The quality of this sample. For raw samples this is an OPC DA2 item quality. For computed samples this is GOOD, BAD or another more precise BAD quality (such as DEVICE FAILURE) taken from the qualities of the raw values used in the computation.

Methods

None.
HistTest.g

HistTest.g — an example script that demonstrates how to access Historian data.

Code

The code for this and other example scripts can be found in the DataHub distribution archive, typically at this location:

C:\Program Files\Cogent\Cogent DataHub\scripts\n
Please refer to the section called “How to Run a Script” for more information on using scripts.

/*
 * This script demonstrates how to access the DataHub Historian data
 * both in raw and processed forms. Generally, a script simply needs
 * to create an instance of the Historian class and then make calls
 * to the methods of that object. See the documentation for the
 * complete list of methods provided.
 *
 * When you query data from the historian the result is a
 * HistoryBuffer instance. This encapsulates the result data in an
 * efficient internal representation that is not visible directly to
 * the script. In order to use the data you must retrieve the data
 * from the HistoryBuffer to produce one or more HistoryValue
 * instances. Each HistoryValue contains a value, a timestamp and a
 * quality. The timestamp is called "xaxis" because in some cases a
 * query could return an y-vs-x result instead of a y-vs-time result.
 * The Historian object does not expose a method for a y-vs-x query,
 * but the underlying implementation will support it.
 *
 * A HistoryBuffer holds its data until there are no more references
 * to the HistoryBuffer object. This means that a HistoryBuffer
 * object that is declared as a local variable in a function will
 * only be valid until the function exits (unless the function
 * returns the HistoryBuffer object). If you need to hold one or
 * more HistoryValue beyond the lifetime of the HistoryBuffer, use
 * copyToArray or copyValue to make a copy of the data.
 *
 * If a history query produces a large amount of data it will be more
 * memory efficient to avoid making a copy of the data. You can
 * access individual HistoryValue instances within the HistoryBuffer
 * using the getValue method. Generally this will take more
 * processing time, but can substantially reduce memory use.
 */
Bear in mind that all scripts run in a single thread. If you spend a long time processing large historical requests, that will disrupt the timing of other scripts.

Statistical queries take substantially longer than raw data queries.

Applications share the execution thread and the global name space, so we create a class that contains all of the functions and variables for the application. This does two things:

1) creates a private name space for the application, and
2) allows you to re-load the application to create either a new unique instance or multiple instances without damaging an existing running instance.

class HistTest Application
{
    historian;
}

method HistTest.rawQuery(pointname, starttime, duration)
{
    local histbuffer, data, datacopy;
    local t1, t2, t3, t4;

    t1 = nanoclock();
    histbuffer = .historian.queryRawData(pointname, starttime, duration);
    t2 = nanoclock();
    data = histbuffer.toArray();
    t3 = nanoclock();
    datacopy = histbuffer.copyToArray();
    t4 = nanoclock();

    pretty_princ("Raw query produced ",
        histbuffer.npoints, " data values\n");
    pretty_princ(" Query took ", t2 - t1, " seconds\n");
    pretty_princ(" Data access took ", t3 - t2, " seconds\n");
Classes from HistorianSupport.g

```java
pretty_princ(" Data copy took ", t4 - t3, " seconds\n");

// If we want to use the values outside this method, we need to
// use a copy, not the raw data. The raw data will be cleaned up
// when the histbuffer object is no longer referenced. We could
// also return the histbuffer object from this method, from which
// we could later query the data using toArray().
datacopy;

}

method HistTest.statisticalQuery(pointname, starttime, duration,
   aggregation_period_secs, statistic)
{
    local histbuffer, data, datacopy;
    local t1, t2, t3, t4, t5, t6, t7;
    local i, numbad, numbad2;

    t1 = nanoclock();
    histbuffer = .historian.queryStatistic(pointname,
        starttime,
        duration,
        aggregation_period_secs,
        statistic);
    t2 = nanoclock();
    data = histbuffer.toArray();
    t3 = nanoclock();
    datacopy = histbuffer.copyToArray();
    t4 = nanoclock();

    pretty_princ("Statistical query, ", statistic, ", produced ",
        histbuffer.npoints, " data values\n");
    pretty_princ(" Query took ", t2 - t1, " seconds\n");
    pretty_princ(" Data access took ", t3 - t2, " seconds\n");
    pretty_princ(" Data copy took ", t4 - t3, " seconds\n");

    // There are a few ways to process the data. We can convert the
    // data to an array, as we do with data and datacopy. We can
    // also walk the data buffer without converting it to an array.
    // The next two examples both do the same thing, walking the data
    // looking for BAD quality.
    numbad = numbad2 = 0;
    t5 = nanoclock();

    // Loop through the data set, having previously converted the
    // buffer data to an array
    with value in data do
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
Classes from HistorianSupport.g

```java
{    if (value.quality == OPC_QUALITY_BAD)
        numbad++;
}
t6 = nanoclock();

// Loop through the data set without converting to an array.
// If the query produces a large amount of data, this will
// certainly be more memory-efficient since a large array does
// not have to be created, but the processing cost of querying
// each value in the data set will be higher.
for (i=histbuffer.npoints-1; i>=0; i--)
{    if (histbuffer.getValue(i).quality == OPC_QUALITY_BAD)
        numbad2++;
}
t7 = nanoclock();

pretty_princ("  Loop through array took ",
t6 - t5, " seconds\n");
   pretty_princ("  Loop through buffer took ",
t7 - t6, " seconds\n");
   pretty_princ("    Found ",
   numbad, " (", numbad2, ") Bad quality values\n");

// Print the last value in the data array, just to show
// some results.
if(array_p(data) && length(data) > 0)
{    local result = data[length(data) -1];
      pretty_princ(" Query results for most recent value: ",
         date_of(result.xaxis), ": ", result.value, "\n\n");
}
}

/* Write the 'main line' of the program here. */
method HistTest.constructor ()
{
   .historian = new Historian();
   princ ("-----------------------------\n");
   .rawQuery("DataPid:PID1.Mv", nanoclock() - 10, 10);
   .rawQuery("DataPid:PID1.Mv", nanoclock() - 100, 100);
   .rawQuery("DataPid:PID1.Mv", nanoclock() - 1000, 1000);
   .rawQuery("DataPid:PID1.Mv", nanoclock() - 10000, 10000);
   princ ("------\n");
   .statisticalQuery("DataPid:PID1.Mv",
```
nanoclock() - 10, 10, 1, "Average");
    .statisticalQuery("DataPid:PID1.Mv",
        nanoclock() - 100, 100, 1, "StDev");
    .statisticalQuery("DataPid:PID1.Mv",
        nanoclock() - 1000, 1000, 1, "Minimum");
    .statisticalQuery("DataPid:PID1.Mv",
        nanoclock() - 10000, 10000, 1, "RegRSquared");
}

/* Any code to be run when the program gets shut down. */
method HistTest.destructor ()
{
}

/* Start the program by instantiating the class. */
ApplicationSingleton (HistTest);
Modbus Commands Methods from ModbusSupport.g

Access to the DataHub's Modbus feature and command set is made available through the ModbusSupport.g script shipped with the DataHub. To use the script, follow these steps:

1. Ensure that the ModbusSupport.g file is present in your Program Files\Cogent\Cogent DataHub\require\ directory.
2. Require the file by adding the following line at the top of your script:
   ```lua
   require (ModbusSupport);
   ```
3. In your script, create a ModbusConfig instance:
   ```lua
   local x = new ModbusConfig();
   ```

All of the methods in ModbusSupport.g are wrappers of a corresponding set of DataHub commands, which are documented in the Cogent DataHub Command Set.
apply

apply — applies all scripted changes.

Synopsis

```
.apply()
```

Description

This method is a wrapper for ModbusApplySettings.
addPoint

addPoint — configures a Modbus point.

Synopsis

```
.addPoint(slaveName, pointName, blockType, pointType, address,
         allowWrite, xformType, xformArgs, conversion)
```

Description

This method is a wrapper for ModbusSlaveAddPoint.
addRange

addRange — configures a range of Modbus points.

Synopsis

```
.addRange(slaveName, pointName, blockType, pointType, address,
    allowWrite, xformType, xformArgs, conversion, itemCount,
    nameStart)
```

Description

This method is a wrapper for ModbusSlaveAddRange.
cancel

cancel — cancels all scripted changes.

Synopsis

.cancel()

Description

This method is a wrapper for ModbusCancelSettings.
createSlave

createSlave — creates a slave connection.

Synopsis

```javascript
.createSlave(slaveName, hostSpec, dataDomain, pollMs, retryMs,
             maxMsgLength, slaveId, supportedFunctions, addressFlags)
```

Description

This method is a wrapper for `ModbusCreateSlave`. 
deletePoint

deletePoint — deletes point connections.

Synopsis

.deletePoint(slaveName, pointNamePattern)

Description

This method is a wrapper for ModbusSlaveDeletePoint.
deleteSlave

deleteSlave — deletes a slave connection.

Synopsis

dele$eSlave(slaveNamePattern)

Description

This method is a wrapper for ModbusDeleteSlave.
enableMaster

enableMaster — enables and disables Modbus master functionality.

Synopsis

```javascript
.enableMaster(enable_p=t)
```

Description

This method is a wrapper for ModbusEnableMaster.
enableSlave

enableSlave — enables and disables Modbus slave connections.

Synopsis

```c
.enableSlave(slaveNamePattern, enable_p?=t)
```

Description

This method is a wrapper for ModbusEnableSlave.
reloadSettings

reloadSettings — loads all Modbus configuration.

Synopsis

```
reloadSettings()
```

Description

This method is a wrapper for ModbusReloadSettings.
slaveExists

slaveExists — checks for the existence of a slave connection.

Synopsis

slaveExists(name)

Description

This method is a wrapper for ModbusQuerySlave.
Appendix A. Basic Troubleshooting

Before doing anything else, you should ensure that the DataHub and Gamma are running properly. You can do this by running the WindowsExample.g file. Start the DataHub, and in the Scripting option of the Properties window, load the WindowsExample.g script. (How do I load a script?) When the script is properly loaded, it should display the Monitor window:

This is a demonstration script for DataHub MS Windows Support. If the DataSim program is running, you should be able to see its data and interact with it through this Monitor. If you cannot, you need to check your DataHub installation and configuration before troubleshooting scripts.

Troubleshooting Scripts

Did the script not run as expected? What happened instead? Here are some suggestions:

• The Script Log displays one or more error messages with relevant line numbers if there are any syntactical or scripting errors, and Gamma will stop executing the script at that point.

• Incorrect output or no output probably means that your script's logic is incorrect. Review your code, correct any errors, and load it again.

• Single expressions, whole lines, or entire blocks of code can be evaluated independently. Highlight the portion of the code text and select Evaluate Selection from the Script menu.

• You can trace the steps of execution by inserting prin statements at strategic points.
Appendix B. License Copyright Information

The following licenses are being used in this product:

License for Scintilla and SciTE

Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
Cogent DataHub® ODBC Scripting

Version 9.0
# Table of Contents

**Introduction** .......................................................................................................................... 1

  - Overview .............................................................................................................. 1

**Setting up a DSN (Data Source Name)** ........................................................................... 1
  - Working with MS Access .................................................................................. 3

**Tutorials** ............................................................................................................................. 4
  - Tutorial 1: Writing new rows to a table, based on a trigger - Multi-Threaded Version .................................................................................................................. 4
  - Tutorial 2: Writing new rows to a table, based on a trigger - Single-Threaded Version .................................................................................................................. 11
  - Tutorial 3: Updating existing rows, or writing new ones ...................................... 17
  - Tutorial 4: Writing data from a database to the DataHub .................................... 22

**Viewing data from a web browser** ................................................................................... 27

**An Explanation of the Tutorial Code** .................................................................................. 29

  - Define the Application Object ............................................................................. 29

**Interactions with the Database** ........................................................................................... 30
  - Connecting to the Database .................................................................................. 30
  - Creating a Gamma Class from a Database Table .................................................. 30
  - Querying Rows from the Database ........................................................................ 31
  - Inserting Rows into a Database ............................................................................. 31
  - Updating Existing Rows in a Database .................................................................. 32
  - Creating a Database Table .................................................................................... 32

**Set up Event Handlers** ........................................................................................................ 33

**Shut Down** .......................................................................................................................... 34

**Multi-Threaded ODBC Interface** ....................................................................................... 36

**How-To** ............................................................................................................................... 36
  - Create an ODBCThread Instance ........................................................................ 36
  - Attach Event Callbacks ......................................................................................... 37
  - Configure Startup Actions .................................................................................... 38
  - Start the Database Thread ...................................................................................... 39

**Store and Forward** ............................................................................................................. 40
  - Time Delayed Writes ............................................................................................. 40

**Example** ................................................................................................................................. 41

**Classes** ................................................................................................................................. 46

  - DATE_STRUCT ........................................................................................................ 47
  - ODBCColumn ......................................................................................................... 48
  - ODBCConnection ................................................................................................. 49
  - ODBCDescriptor ................................................................................................. 52
  - ODBCEnvironment ............................................................................................... 53
  - ODBCHandle ......................................................................................................... 54
  - ODBCResult .......................................................................................................... 55
  - ODBCStatement ................................................................................................... 56
  - ODBCThread ......................................................................................................... 58
  - ODBCThreadResult ............................................................................................. 65
  - SQLGUID ................................................................................................................. 67
GLOBAL FLOATING TYPES

SQL_DAY_SECOND_STRUCT ................................................................. 68
SQL_INTERVAL_STRUCT ..................................................................... 69
SQL_INTERVAL_STRUCT_intval ............................................................ 70
SQL_NUMERIC_STRUCT ..................................................................... 71
SQL_YEAR_MONTH_STRUCT ............................................................... 72
TIMESTAMP_STRUCT ....................................................................... 73
TIME_STRUCT ................................................................................ 74

Global Functions .................................................................................. 75
ODBC_AllocEnvironment ....................................................................... 76
ODBC_ValueString ................................................................................ 77

Constants .............................................................................................. 78
Introduction

Overview

The Cogent DataHub has built-in scripting capabilities (see DataHub Scripting), which among other things let you connect the DataHub to any ODBC-compliant database. This guide assumes a basic understanding of DataHub scripting, and provides the specific information you will need to script connections between the DataHub and an ODBC database.

DataHub ODBC scripting uses wrapped MSDN functions. Therefore, this documentation often links to and refers to the MSDN documentation. You may need to conduct a search in that documentation to find the corresponding function.

The tutorials in this manual use SQL commands to query the database. The syntax for these commands may vary slightly from one ODBC database to another. If a given tutorial doesn't work right away, check the syntax of the SQL commands in the tutorial against the syntax that your database uses.

Although DataHub Scripting is included with every Cogent DataHub license, ODBC scripting requires that the Database feature be licensed as well.

Setting up a DSN (Data Source Name)

To connect an ODBC-compliant database to the DataHub, you will need to ensure that you have specified a DSN (Data Source Name) for the database. Here’s how:

1. From the Windows Start menu, choose Control Panel, then Administrative Tools, and then Data Sources (ODBC) to open the ODBC Data Source Administrator window. This is what it looks like in Windows XP:
2. Select the **User DSN** or **System DSN** tab, depending on how you plan to access your database.

   A user DSN is only available to the current user account, while a system DSN is available to any user account on the computer.

3. Now you can add a new database or configure an existing one.

**Add a new database**

1. Click the **Add** button. The Create New Data Source window will open, displaying a list of data source drivers.

2. Select the data source driver that corresponds to your ODBC database. A data source setup window will open. Each data source setup window is different, but you should be able to find the appropriate entry fields easily enough.

3. Enter the data source name and select the database.

4. Enter any other required or optional information such as login name, password, etc. What entries need to be made and where they are entered depends on the particular data source setup window you are using.

5. Click **OK** to return to the ODBC Data Source Administrator window. You should be able to see the new database and driver listed. If you need to make any changes, you can configure an exiting database, as explained below.

**Configure an existing database**

1. Select a data source name and click the **Configure...** button. This takes you to the data source setup window (explained above) where you can make changes to the configuration.

2. Make your changes and click **OK** to return to the ODBC Data Source Administrator window. Any time you need to make a change, you can go to this window.
4. When you are satisfied everything is correct, click the **OK** button to exit the ODBC Data Source Administrator.

**Working with MS Access**

The Microsoft Access database program is a handy tool for MS Office users, but it is not completely ODBC compliant, nor is it a database server. Its design prevents simultaneous updates from outside data sources while the Access program is running, but it can still be used with the DataHub to collect and store data from real-time systems.

**File-based Data Access**

MS Access is not a database server like MS SQL Server, MySQL, Oracle, and others. Instead, it accesses a data file (.mdb file), reading from and writing data to that file. Other programs like the DataHub can also access the file, but *not simultaneously* with Access. You can use the DataHub to modify the data file, but any time you open the file in Access, all programs including the DataHub are blocked from using it until you close the file in Access.

What this means is that if you are using the DataHub to interface to a real-time control or financial system and you want mission critical data stored in an ODBC-compliant database, you probably don’t want to be using MS Access. However, it could be useful for storing and viewing logged data. And for some it might be a convenient way to start investigating the possibilities of ODBC scripting with the DataHub.

**Queries on Primary Keys**

MS Access does not support the ODBC function **SQLPrimaryKeys**, which means you cannot programatically discover the primary keys of an Access database. Thus you will need to identify the primary keys of the tables in the code itself. You will notice in our tutorials that we do just that. Since data table design doesn’t change frequently, this should not prove to be a problem in most cases.
Tutorials

Tutorial 1: Writing new rows to a table, based on a trigger - Multi-Threaded Version

This script creates and inserts a new row into a database whenever a trigger point changes value. The data that gets inserted into the row is an ID for the entry (the primary key), the name of a specified point, its value, and a timestamp of the change. The script uses the multi-threaded feature Store and Forward to store data in memory and/or on disk if the database is disconnected or too busy, and then log that data in time-sequential order when the database is available again.

The tutorials in this manual use SQL commands to query the database. The syntax for these commands may vary slightly from one ODBC database to another. If a given tutorial doesn't work right away, check the syntax of the SQL commands used here against the syntax that your database uses.

Although DataHub Scripting is included with every Cogent DataHub license, ODBC scripting requires that the Cogent DataHub Database feature be licensed as well.

Getting Started

To run this code or the other tutorials in this manual, you will need to do the following:

1. Set up a DSN (Data Source Name) called "DataHubThreadedTest" and point it to an empty database on your database server.
2. Create a table in the database named "datatable" that contains at least four columns with names, data types, and other attributes exactly as specified here:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Other attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptid</td>
<td>integer</td>
<td>identity, non-null, counter</td>
</tr>
<tr>
<td>ptname</td>
<td>text string</td>
<td>null</td>
</tr>
<tr>
<td>ptvalue</td>
<td>real</td>
<td>non-null</td>
</tr>
<tr>
<td>pttime</td>
<td>datetime</td>
<td>null</td>
</tr>
</tbody>
</table>

Any other columns in this table must be allowed to take on a null value.

3. Start DataSim.
4. Find the tutorial script ODBCTutorial1.g on your system, and run it.

You can access DataHub scripts and scripting capabilities by pressing the Scripting button in the Properties window, to display the Scripting and Customization screen. The upper half of the screen shows the Gamma files currently configured in the DataHub.
5. Check the database table to see the results. Once you have it working, you can modify the code as explained below.

**The Code: ODBCTutorial1.g**

```javascript
/*
 * This script demonstrates the use of the threaded ODBC interface
 * to insert data from the DataSim program into a database based on
 * a timer or an event.
 */

require ("Application");
require ("ODBCThreadSupport");
require ("Time");
```
require ("Quality");

class ODBCTutorial1 Application
{
    DSN = "MySQLLocal";  // The DSN name to use for the database
    // connection
    username = "test";   // The user name for connecting to
    // the database
    password = "test";   // The password for connecting to
    // the database
    tablename = "test";  // The name of the database table
    cachefile = "c:/tmp/testcache.txt"; // Base name for the disk
    // cache file
    tableclass;
    thread;
}

/* This method will be called every time the connection is
 * established to the database. If there is something we only want to
 * perform on the first connection, we can test is_first_connect to
 * perform the code only once.
 */
method ODBCTutorial1.onConnect()
{
    princ ("Connection succeeded\n");
    if (.thread.is_first_connect)
    {
        // Start the sequence defined by the AddInitStage calls
        // in the constructor
        .thread.BeginAsyncInit();
    }
}

/* If we get a connection attempt failure, or the connection fails
 * after having been connected, this method is called.
 */
method ODBCTutorial1.onConnectFail()
{
    princ ("Connection closed: ", SQLResult.Description, "\n");
}

/* Map the table in the set of table definitions that matches the
 * name in .tablename into a Gamma class. This lets us easily
 * convert between class instances and rows in the table.
 */
method ODBCTutorial1.mapTable(name, tabledefinitions)
{  .tableclass = .thread.ClassFromTable(name, tabledefinitions);
}

/* Set up the timer or event handler functions to write to  
* the table. */
method ODBCTutorial1.startLogging()
{
    /* You can modify and/or add similar timers or event handlers  
    * for each data point that you want to log. Please refer to  
    * the "Methods and Functions from Application.g" section of  
    * the documentation for more details about the timer and event  
    * handler functions. */
    // Log a new row of data every 3 seconds.
    .TimerEvery(3, `(@self).writeData(#$DataSim:Sine));

    // Log a new row of data at 20 seconds past each minute of  
    // each hour, etc.
    .TimerAt(nil, nil, nil, nil, nil, 20,  
      `(@self).writeData(#$DataSim:Triangle));

    // Log a new row of data for the point DataSim:Square when it  
    // changes.
    .OnChange(#$DataSim:Square, `(@self).writeData(this));

    // Log a new row of data for the point DataSim:Sine when  
    // DataSim:Square changes.
    .OnChange(#$DataSim:Square, `(@self).writeData(#$DataSim:Sine));
}

method ODBCTutorial1.writeData(pointsymbol)
{
    local       row = new (.tableclass);
    local       pttime, ptltime;
    local       timestring;

    /* Generate a timestamp in database-independent format to  
    * the millisecond. Many databases strip the milliseconds from  
    * a timestamp, but it is harmless to provide them in case the  
    * database can store them. */
    pttime =  
        WindowsTimeToUnixTime(PointMetadata(pointsymbol).timestamp);
    ptltime = localtime(pttime);

    ...
timestring = format("{ts '%04d-%02d-%02d %02d:%02d:%02d.%03d'}",
    ptlttime.year+1900, ptlttime.mon+1,
    ptlttime.mday, ptlttime.hour, ptlttime.min,
    ptlttime.sec,
    (ptlttime % 1) * 1000);

    /* Fill the row. Since we mapped the table into a Gamma class, */
    /* we can access the columns in the row as member variables of */
    /* the mapped class. */
    /* */
    row.ptname = string(pointsymbol);
    row.ptvalue = eval(pointsymbol);
    row.pttime = timestring;
    /* Perform the insertion. In this case we are providing no */
    /* callback on completion. */
    /* */
    .thread.Insert(row, nil);
}

/* Write the 'main line' of the program here. */
method ODBCTutorial1.constructor ()
{

    // Create and configure the database connection object
    .thread = new ODBCThread();
    .thread.Configure(.DSN, .username, .password,
        STORE_AND_FORWARD, .cachefile, 0);

    /* Use this to delete the table on the first connection after */
    /* the script starts. */
    /* BE CAREFUL - re-running the script will start over and delete */
    /* the table again. */
    /* */
    // .thread.AddInitStage(format("drop table %s", .tablename),
    //     nil, t);

    /* Use this to create the table if it does not exist. */
    /* Note: this might not work for all databases. */
    /* When in doubt, create the table manually. The 't' in the */
    /* onFail argument says to ignore errors and continue with the */
    /* next stage. */
    /* */
    .thread.AddInitStage(format("create table %s (ptid int
        auto_increment primary key, ptname
        varchar(64), ptvalue double, pttime
        datetime )", .tablename), nil, t);
/* Query the table and map it to a class for each insertion. We want to run an asynchronous event within the asynchronous initialization stage, so to do that we specify the special method cbInitStage as the callback function of our asynchronous event (GetTableInfo). We deal with the return from the GetTableInfo in the onSuccess argument of the init stage. */

.thread.AddInitStage(`(@.thread).GetTableInfo("", "",
  (@.tablename),
  "TABLE,VIEW",
  `(@.thread)
    .cbInitStage()),
  `(@self).mapTable(@.tablename, SQLTables), nil);

/* Do not start writing data to the table until we have successfully created and mapped the table to a class. If we wanted to start writing data immediately, then we would create the table class beforehand instead of querying the database for the table definition. Then, even if the database were unavailable, we could still cache to the local disk until the database was ready. */

.thread.AddInitStage(nil, `(@self).startLogging(), nil);

/* Set up the callback functions for various events from the database thread */

.thread.OnConnectionSucceeded = `(@self).onConnect();
.thread.OnConnectionFailed = `(@self).onConnectFail();
.thread.OnFileSystemError = `princ("File System Error: ", SQLResult, "\n");
.thread.OnODBCError = `princ("ODBC Error: ", SQLResult, "\n");
.thread.OnExecuteStored = nil;

/* Now that everything is configured, start the thread and begin connecting. All of the logic will be driven through the onConnect callback and then through the init stages. */

.thread.Start();

/* Create a menu item in the system tray that allows us to open a window to monitor the performance of the ODBC thread. The menu strings can be edited as desired. */
.AddCustomSubMenu("ODBC Thread Demo");
.AddCustomMenuItem("Monitor Performance",
  `( thread).CreateMonitorWindow(( self),
  "ODBC Demo Monitor");

/* If we want to open the performance monitor window when the
 * script starts, do it here.
 */
.thread.CreateMonitorWindow(self, "ODBC Demo Monitor");

/* Any code to be run when the program gets shut down. */
.method ODBCTutorial1.destructor ()
{
  if (instance_p (.thread))
    destroy (.thread);
}

/* Start the program by instantiating the class. */
.ApplicationSingleton (ODBCTutorial1);

Modifying the Code

You can modify the startLogging method to add your own points by replacing data domains (domain), point names (point) and/or times (day, month, year, hour, minute, second, etc.) like this:

```csharp
// Log a new row of data every # seconds.
.TimerEvery(seconds, `(@self).writeData(#$domain:point));

// Log a new row of data at # seconds past each minute of each
// hour, etc.
.TimerAt(day, month, year, hour, minute, second,
  `(@self).writeData(#$domain:point));

// Log a new row of data for a point when it changes.
.OnChange(#$domain:point, `(@self).writeData(this));

// Log a new row of data for a point when a trigger point changes.
.OnChange(#$domain:point, `(@self).writeData(#$domain:point));
```

Please refer to the documentation for these methods of the Application class for more information: TimerEvery, TimerAt, and OnChange.
Tutorial 2: Writing new rows to a table, based on a trigger - Single-Threaded Version

This script creates and inserts a new row into a database whenever a trigger point changes value. The data that gets inserted into the row is an ID for the entry (the primary key), the value of a specified point, the timestamp of the change, and the name and quality of the point. The script also checks the connection to the database, and will attempt to reconnect every 5 seconds if the connection is lost.

The tutorials in this manual use SQL commands to query the database. The syntax for these commands may vary slightly from one ODBC database to another. If a given tutorial doesn't work right away, check the syntax of the SQL commands used here against the syntax that your database uses.

Although DataHub Scripting is included with every Cogent DataHub license, ODBC scripting requires that the Cogent DataHub Database feature be licensed as well.

Getting Started

To run this code or the other tutorials in this manual, you will need to do the following:

1. Set up a DSN (Data Source Name) called "DataHubTest" and point it to an empty database on your database server.
2. Create a table in the database named "datatable" that contains at least five columns with names, data types, and other attributes exactly as specified here:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Other attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>integer</td>
<td>identity, non-null, counter</td>
</tr>
<tr>
<td>PTVALUE</td>
<td>real</td>
<td>non-null</td>
</tr>
<tr>
<td>PTTIME</td>
<td>datetime</td>
<td>null</td>
</tr>
<tr>
<td>PTNAME</td>
<td>text string</td>
<td>null</td>
</tr>
<tr>
<td>PTQUALITY</td>
<td>text string</td>
<td>null</td>
</tr>
</tbody>
</table>

Any other columns in this table must be allowed to take on a null value.

3. Start DataSim.
4. Find the tutorial script ODBCTutorial2.g on your system, and run it.

You can access DataHub scripts and scripting capabilities by pressing the Scripting button in the Properties window, to display the Scripting and Customization screen. The upper half of the screen shows the Gamma files currently configured in the DataHub:
The **Open** button opens a file selector for you to add an existing script to the list. Scripts are kept in a DataHub’s **scripts** folder. Scripts that come with the DataHub are installed here (32-bit or 64-bit):

```
C:\Program Files (x86)\Cogent\Cogent DataHub\scripts\nC:\Program Files\Cogent\Cogent DataHub\scripts\n```

All content in this directory will be replaced by the default content when the DataHub is re-installed. If you plan to edit one of these scripts, or to write a new script, you should keep it in this folder for user-created scripts:

```
C:\Users\Username\AppData\Roaming\Cogent DataHub\scripts
```

The **Edit** button opens the selected script in the **Script Editor** for editing.

You can view error messages and printed output from a script in the **Script Log**. To open the Script Log, right click on the DataHub icon in the system tray, and select **View Script Log**.

For complete information about DataHub scripting, please refer to the **DataHub Scripting Manual**

5. Check the database table to see the results. Once you have it working, you can **modify the code** as explained below.

**The Code: ODBCTutorial2.g**

```plaintext
/* All user scripts should derive from the base "Application" class */
require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or
 * Windows programming. Uncomment either or both. */

//require ("WindowsSupport");
require ("ODBCSupport");
```

/**
 * Applications share the execution thread and the global name space, so we create a class that contains all of the functions and variables for the application. This does two things:
 * 1) creates a private name space for the application, and
 * 2) allows you to re-load the application to create either a new unique instance or multiple instances without damaging an existing running instance.
 */

/**
 * This application assumes that the table specified by the "tablename" member variable exists in the DSN specified by the "DSN" member variable below.
 * The table consists of at least the following columns:
 * ID - integer, identity, non-null, counter
 * PTVALUE - real, non-null
 * PTTIME - datetime, null
 * PTNAME - text string, null
 * PTQUALITY - text string, null
 * Any other columns in this table must be allowed to take on a NULL value.
 */

class ODBCTutorial2 Application
{
    /* User-defined values, may be changed as needed. */
    DSN = "DataHubTest";
    user = "test";
    password = "test";
    tablename = "datatable";

    /* These values get defined by the program. */
    conn;
    env;
    tableclass;
    is_connected;
    is_connecting;
}

/* Connect to the DSN and create a class that maps the table. */
method ODBCTutorial2.Connect()
{
    princ ("Connecting to database
");
}
.is_connecting = t;

protect
{
    /* Create the ODBC environment and connection */
    if (!.env)
        .env = ODBC_AllocEnvironment();
    if (!.conn)
        .conn = .env.AllocConnection();

    /* Attempt the connection. */
    ret = .conn.Connect (.DSN, .user, .password);
    if (ret != SQL_SUCCESS && ret != SQL_SUCCESS_WITH_INFO)
        error (.conn.GetDiagRec());

    /* Create a class from the table */
    .tableclass = .conn.ClassFromTable (#DataEntry, nil, .tablename);

    /* Set the primary key. This is redundant for MS-SQL and
    MYSQL since they can figure it out themselves, but Access
    requires it. */
    mykey = .conn.SetPrimaryKey (.tableclass, "ID");

    .is_connected = t;
}
unwind
{
    .is_connecting = nil;
    if (.is_connected)
        princ (" Connection successful\n");
    else
        princ (" Connection failed\n");
}

/* Disconnect from the database server */
method ODBCTutorial2.Disconnect ()
{
    if (.conn)
    {
        try
        {
            princ ("Disconnecting database\n");
            .conn.Disconnect();
            destroy (.conn);
        }
        catch
        {  
            princ ("Error disconnecting database\n");
            .conn.Disconnect();
            destroy (.conn);
        }
    }
}
catch
{
    princ ("Disconnection failed: ", _last_error_, "\n");
}
.conn = nil;
.is_connected = nil;

method ODBCTutorial2.Reconnect()
{
    if (!.is_connected && !.is_connecting)
    {
        .Connect();
    }
}

/* Fill a database record with new information from a point change. */
method ODBCTutorial2.FillRecord (record, sym, newvalue)
{
    local   timestamp;
    timestamp = localtime (PointGetUnixTime(sym));
    timestamp = format ("%d-%02d-%02d %02d:%02d:%02d",
        timestamp.year + 1900, timestamp.mon + 1,
        timestamp.mday, timestamp.hour, timestamp.min,
        timestamp.sec);
    record.PTNAME = string (sym);
    record.PTVALUE = number (newvalue);
    record.PTTIME = timestamp;
    record.PTQUALITY = GetQualityName(PointMetadata(sym).quality);
    record;
}

/* Write a new record into the database based on a point change. */
method ODBCTutorial2.AddRecord (sym, newvalue)
{
    local    record = new (.tableclass);
    .FillRecord (record, sym, newvalue);
    try
    {
        .conn.Insert (record);
    }
    catch
    {

princ
("Write failed. Disconnecting. Record was not written.\n");
  .Disconnect();
}
record;
}

/* The mainline. Connect to the database and begin storing data from
the DataHub into the database. */
method ODBCTutorial2.constructor ()
{
  local    ret;

  /* Start a timer that will reconnect to the database
   every 5 seconds if the connection has been lost. */
  .TimerEvery(5, `(@self).Reconnect());

  /* Try connecting now. If this fails, the timer will
   try again later. */
  .Reconnect();

  /* Add a record when a point changes. */
  .OnChange (#$DataSim:Square,
    `(@self).AddRecord (#$DataSim:Sine, $DataSim:Sine));

  /* Add more points like this:
   * .OnChange (#$DataSim:Square,
   * Have the trigger point's value get written like this:
   * .OnChange (#$DataSim:Square,
   *   `(@self).AddRecord
   *     (#$DataSim:Square, $DataSim:Square));
   */
}

/* Any code to be run when the program gets shut down. */
method ODBCTutorial2.destructor ()
{
  .Disconnect();
  if (.env)
    destroy(.env);
}

/* Start the program by instantiating the class. If your
* constructor code does not create a persistent reference to
the instance (self), then it will be destroyed by the
garbage collector soon after creation. If you do not want
this to happen, assign the instance to a global variable, or
create a static data member in your class to which you assign
'self' during the construction process. ApplicationSingleton()
does this for you automatically. */
ApplicationSingleton (ODBCTutorial2);

Modifying the Code

There are several ways you can modify the code. These modifications are made in the
ODBCTutorial1.constructor method, towards the end of the script.

• **Add more DataSim points** using this format:

```plaintext
.OnChange (#$DataSim:Square,
  `(self).AddRecord (#$DataSim:pointname,
 $DataSim:pointname));
```

Where `pointname` is the name of a point in DataSim.

• **Change the trigger point** like this:

```plaintext
.OnChange (#$DataSim:pointname,
  `(self).AddRecord (#$DataSim:Sine, $DataSim:Sine));
```

Where `pointname` is the name of a point in DataSim.

• **Add your own points** You can add your own points using this syntax:

```plaintext
.OnChange (#$domain:pointname,
  `(self).AddRecord (#$domain:pointname,
 $domain:pointname));
```

where `domain` is the domain that the point is in, and `pointname` is the name of the
point.

**Tutorial 3: Updating existing rows, or writing new ones**

This tutorial demonstrates how to find a particular row and update it, as well as write new
rows, depending on the point.

This tutorial uses the same DSN, database, and table as Tutorial 2. If you
haven't done Tutorial 2 yet, please review Getting Started in that section to see
how to set up your system for this tutorial.

Although DataHub Scripting is included with every Cogent DataHub license,
ODBC scripting requires that the Cogent DataHub Database feature be li-
censed as well.
The Code: ODBCTutorial3.g

```c
/* All user scripts should derive from the base "Application" class */
require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or
 * Windows programming. Uncomment either or both. */
//require ("WindowsSupport");
require ("ODBCSupport");
require ("Time");
require ("Quality");

/* Applications share the execution thread and the global name
 * space, so we create a class that contains all of the functions
 * and variables for the application. This does two things:
 * 1) creates a private name space for the application, and
 * 2) allows you to re-load the application to create either
 *    a new unique instance or multiple instances without
 *    damaging an existing running instance. */

/*
 * This application assumes that the table specified by the
 * "tablename" member variable exists in the DSN specified by the
 * "DSN" member variable below.
 * The table consists of at least the following columns:
 * ID - integer, identity, non-null, counter
 * PTVALUE - real, non-null
 * PTTIME - datetime, null
 * PTNAME - text string, null
 * PTQUALITY - text string, null
 * Any other columns in this table must be allowed to take on a
 * NULL value.
 */

class ODBCTutorial3 Application
{
    /* User-defined values, may be changed as needed. */
    DSN = "DataHubTest";
    user = "test";
    password = "test";
    tablename = "datatable";
}
```
/* These values get defined by the program. */
conn;
env;
tableclass;
}

/* Connect to the DSN and create a class that maps the table. */
method ODBCTutorial3.Connect ()
{
  /* Create the ODBC environment and connection */
  .env = ODBC_AllocEnvironment();
  .conn = .env.AllocConnection();

  /* Attempt the connection. */
  ret = .conn.Connect (.DSN, .user, .password);
  if (ret != SQL_SUCCESS && ret != SQL_SUCCESS_WITH_INFO)
    error (.conn.GetDiagRec());

  /* Create a class from the table */
  .tableclass = .conn.ClassFromTable (#DataEntry, nil, .tablename);

  /* Set the primary key. This is redundant for MS-SQL and MYSQL 
   * since they can figure it out themselves, but Access requires 
   * it. */
  mykey = .conn.SetPrimaryKey (.tableclass, "ID");
}

/* Fill a database record with new information from a point change */
method ODBCTutorial3.FillRecord (record, sym, newvalue)
{
  local  timestamp;
  timestamp = localtime (PointGetUnixTime(sym));
  timestamp = format ("%d-%02d-%02d %02d:%02d:%02d",
    timestamp.year + 1900, timestamp.mon + 1,
    timestamp.mday, timestamp.hour, timestamp.min,
    timestamp.sec);
  record.PTNAME = string (sym);
  record.PTVALUE = number (newvalue);
  record.PTTIME = timestamp;
  record.PTQUALITY = GetQualityName(PointMetadata(sym).quality);
  record;
}

/* Write a new record into the database based on a point change. */
method ODBCTutorial3.AddRecord (sym, newvalue)
{  
  local record = new DataEntry();  
  .FillRecord (record, sym, newvalue);  
  .conn.Insert (record);  
  record;  
}

/* Write a data point into a field of a record. This is called from a DataHub point change event. This method will replace an existing record that is cached with the point at startup. If there was no existing row in the database, this will create one and then update it in subsequent calls. */
method ODBCTutorial3.UpdateRecord (sym, newvalue)
{
  local record = getprop (sym, #dbrecord);
  if (!record)
  {
    record = .AddRecord (sym, newvalue);
    setprop (sym, #dbrecord, record);
  }
  else
  {
    .FillRecord (record, sym, newvalue);
  .conn.Update (record);
  }
}

/* Find an existing record in the database for this point. If it exists, associate the record with the point. */
method ODBCTutorial3.GetExistingRecord (sym, klass)
{
  local result = .conn.QueryToClass (klass, string
     ("SELECT * FROM ",
      klass.__table,
      " WHERE PTNAME = ",
      sym, ",
      "));
  if (array_p(result))
    setprop (sym, #dbrecord, result[0]);
}

/* Start updating the database whenever a point changes. If the overwrite argument is non-nil or absent, then this method will cause an existing record in the database to be overwritten each time. If overwrite is nil, every point change will create a new row in the database. */
method ODBCTutorial3.WatchPoint (sym, tableclass, overwrite?=t)
{ /* Grab an existing record for this point if it exists */
  .GetExistingRecord (sym, tableclass);
  if (overwrite)
    .OnChange (sym, `(@self).UpdateRecord (this, value));
  else
    .OnChange (sym, `(@self).AddRecord (this, value));
}

/* The mainline. Connect to the database and begin storing data from
the DataHub into the database. */
method ODBCTutorial3.constructor ()
{
  local ret;

  /* Connect to the DSN. */
  .Connect();

  /* Register points that we want to save. The WatchPoint method
  takes an optional third argument. If it is nil, every point
  change will add a row to the table. If it is absent or
  non-nil, then every point change overwrites the existing row
  in the table for that point. */
  .WatchPoint (#$DataSim:Square, .tableclass, nil);
  .WatchPoint (#$DataSim:Sine, .tableclass);
}

/* Any code to be run when the program gets shut down. */
method ODBCTutorial3.destructor ()
{
}

/* Start the program by instantiating the class. If your
* constructor code does not create a persistent reference to
* the instance (self), then it will be destroyed by the
* garbage collector soon after creation. If you do not want
* this to happen, assign the instance to a global variable, or
* create a static data member in your class to which you assign
* 'self' during the construction process. ApplicationSingleton()
* does this for you automatically. */
ApplicationSingleton (ODBCTutorial3);
Modifying the Code

There are several ways you can modify the code. These modifications are made in the ODBCTutorial2.constructor method, towards the end of the script.

• **Overwrite or add rows.** There are two ways the ODBC database can receive the data: by overwriting old values with new values in a single row, or by adding a new row for each new value. These are determined by the last argument in the .WatchPoint function:

  ```
  .WatchPoint ($DataSim:Square, tableclass, nil);
  .WatchPoint ($DataSim:Sine, tableclass);
  ```

  The default is to overwrite values. This is what happens for values pertaining to DataSim:Sine. To have the DataHub write a new line for each change, you can add a final argument, nil, such as in DataSim:Square above.

• **Add more DataSim points.** To add other points from DataSim, use this format for new rows:

  ```
  .WatchPoint ($DataSim:pointname, tableclass, nil);
  ```

  or this format for overwriting rows:

  ```
  .WatchPoint ($DataSim:pointname, tableclass);
  ```

  Where `pointname` is the name of a point in DataSim.

• **Add your own points** You can add your own points using this syntax:

  ```
  .WatchPoint ($domain:pointname, tableclass, nil);
  .WatchPoint ($domain:pointname, tableclass);
  ```

  where `domain` is the domain that the point is in, and `pointname` is the name of the point.

**Tutorial 4: Writing data from a database to the DataHub**

This tutorial demonstrates how to keep the DataHub updated every second with the latest values in a database.

This tutorial uses the same DSN and database as Tutorial 2, but creates a different table called "control" (see below). If you haven't done Tutorial 2 yet, please review Getting Started in that section to see how to set up your system for this tutorial.

Although DataHub Scripting is included with every Cogent DataHub license, ODBC scripting requires that the Cogent DataHub Database feature be licensed as well.

As with Tutorial 2, you will need to create a table in the database. This new table should be named "control", and should contain at least three columns with names, data types, and other attributes exactly as specified here:
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Other attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>integer</td>
<td>identity, non-null, counter</td>
</tr>
<tr>
<td>CTRLNAME</td>
<td>text string</td>
<td>null</td>
</tr>
<tr>
<td>CTRLVALUE</td>
<td>real</td>
<td>non-null</td>
</tr>
</tbody>
</table>

Any other columns in this table must be allowed to take on a null value.

Once this script is running, you can enter the name of any existing DataHub point in a row of the database in the **CTRLNAME** column. Make sure you enter the full point name, including the domain name, with the syntax *domainname:pointname*. Enter a corresponding value for the point in **CTRLVALUE**. The entered value will appear for that point in the DataHub. Any time the value changes in the database, the results get passed to the DataHub within a second. The point in the DataHub will continue to be updated once every second from the database as long as the two are both running.

The Code: `ODBCTutorial4.g`

```sql
/* All user scripts should derive from the base "Application" class */
require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or
* Windows programming. Uncomment either or both. */
require ("ODBCSupport");

/* Applications share the execution thread and the global name
* space, so we create a class that contains all of the functions
* and variables for the application. This does two things:
* 1) creates a private name space for the application, and
* 2) allows you to re-load the application to create either
* a new unique instance or multiple instances without
* damaging an existing running instance. */

/* This application assumes that the table specified by the
* "tablename" member variable exists in the DSN specified by the
* "DSN" member variable below. The table consists of at least the following columns:
* ID - integer, identity, non-null, counter
* CTRLNAME - text string, null
* CTRLVALUE - real, non-null
* Any other columns in this table must be allowed to take on a
* NULL value.
```
class ODBCTutorial4 Application
{
    /* User-defined values, may be changed as needed. */
    DSN = "DataHubTest";
    user = "test";
    password = "test";
    tablename = "Table1";

    /* These values get defined by the program. */
    conn;
    env;
    tableclass;
    is_connected;
    is_connecting;
}

/* Connect to the DSN and create a class that maps the table. */
method ODBCTutorial4.Connect ()
{
    local ret;
    .is_connecting = t;

    protect
    {
        /* Create the ODBC environment and connection */
        if (!.env)
            .env = ODBC_AllocEnvironment();
        if (!.conn)
            .conn = .env.AllocConnection();

        /* Attempt the connection. */
        ret = .conn.Connect (.DSN, .user, .password);
        if (ret != SQL_SUCCESS && ret != SQL_SUCCESS_WITH_INFO)
            error (.conn.GetDiagRec());

        /* Create a class from the table */
        .tableclass = .conn.ClassFromTable (#DataEntry, nil, .tablename);

        /* Set the primary key. This is redundant for MS-SQL and
        MYSQL since they can figure it out themselves, but Access
        requires it. */
        .conn.SetPrimaryKey (.tableclass, "ID");
    }
}
.is_connected = t;
}
unwind
{
    .is_connecting = nil;
    if (.is_connected)
        princ (" Connection successful\n");
    else
        princ (" Connection failed\n");
}

/* Disconnect from the database server */
method ODBCTutorial4.Disconnect ()
{
    if (.conn)
    {
        try
        {
            princ ("Disconnecting database\n");
            .conn.Disconnect();
            destroy (.conn);
        }
        catch
        {
            princ ("Disconnection failed: ", _last_error_, "\n");
        }
        .conn = nil;
    }
    .is_connected = nil;
}

/* Try to reconnect to the database if it's not currently connected. */
method ODBCTutorial4.Reconnect ()
{
    if (!.is_connected && !.is_connecting)
    {
        .Connect();
    }
}

/* Reload information from a database record into the DataHub. This is being called on a timer. We re-query the database for the given record, then update the DataHub points simply by assigning them. */
method ODBCTutorial4.Update ()
{
    local result;

    if (.is_connected)
    {
        try
        {
            result = .conn.QueryToClass (.tableclass,
            string ("select * from ", .
            tablename));
            with x in result do
            {
                datahub_write (x.CTRLNAME, x.CTRLVALUE);
            }
        }
        catch
        {
            /* If the query fails then disconnect and do not try
            again until a successful connection has been made
            based on the reconnect timer. */
            princ("Query failed. Disconnecting - ",
            _last_error_, "\n");
            .Disconnect();
        }
    }
}

/* The mainline. Connect to the database and begin storing data from
the DataHub into the database. */
method ODBCTutorial4.constructor ()
{
    /* Every second, read the 'control' database, and update the
values. This keeps the value in the DataHub always in sync
with the database. The timer value can be fractional, such
as 0.5 for twice per second. */
    .TimerEvery (1, `(@self).Update ());

    /* Start a timer that will reconnect to the database
every 5 seconds if the connection has been lost. */
    .TimerEvery(5, `(@self).Reconnect());

    /* Try connecting now. If this fails, the timer will try again
later. */
    .Reconnect();
/* Any code to be run when the program gets shut down. */
method ODBCTutorial4.destructor ()
{
    .Disconnect();
    if (.env)
        destroy(.env);
}

/* Start the program by instantiating the class. If your
* constructor code does not create a persistent reference to
* the instance (self), then it will be destroyed by the
* garbage collector soon after creation. If you do not want
* this to happen, assign the instance to a global variable, or
* create a static data member in your class to which you assign
* 'self' during the construction process. ApplicationSingleton()
* does this for you automatically. */
ApplicationSingleton (ODBCTutorial4);

Viewing data from a web browser

Using the DataHub Web Server, it is possible to send an SQL query from a web page to a
database, and view the results in the web page.
Please refer to the Web Server documentation in the Cogent DataHub manual for more information.
An Explanation of the Tutorial Code

The DataHub interacts with relational databases through ODBC (Open Data Base Connectivity). Every database acts a little differently, and each person's requirements are specific to their own application. Consequently, the DataHub offers its ODBC support through scripts, written in the Gamma language.

A DataHub script consists of four parts:

- Define the application object
- Interactions with the database
- Set up event handlers
- Shut down

Define the Application Object

A DataHub script defines an object called an **Application** object. This object is a class derived from the class **Application**. All of the functions that you define should be methods of this class. Variables should be members of this class wherever possible.

```gamma
class MyODBCScript Application
{
    env;    // store the ODBC environment here
    conn;   // store the ODBC connection here
}
```

This defines the application object.

The application object requires a constructor. The constructor acts as the main line of your program. To start your program, you just create an instance of the application class, causing the constructor to run.

```gamma
method MyODBCScript.constructor ()
{
    // This is the program main line
}
```

The job of the constructor is to define event handlers. An event handler is program code attached to a particular event. An event can be one of:

- A change in a DataHub point value
- A timer
- A Microsoft Windows event

The constructor runs to completion. Once the constructor completes, the script engine begins processing events, and executes the program code attached to those events.
The script engine will continue to process events until the application object is destroyed. You may optionally provide a destructor for the object to clean up any specific data or open any files or timers that your application has created.

```cpp
method MyODBCScript.destructor ()
{
    // This is where you clean up open files and connections
}
```

You can add other methods to the application object as you need them. Only the constructor is necessary.

**Interactions with the Database**

**Connecting to the Database**

Connecting to an ODBC database is a three-step process:

1. Create an ODBCEnvironment object.
2. Create an ODBCConnection object.
3. Connect to a DSN (Data Source Name).

```cpp
.env = ODBC_AllocEnvironment();
.conn = .env.AllocConnection();
/* Attempt to connect. */
ret = .conn.Connect ("DSN name", "user name", "password");
if (ret != SQL_SUCCESS && ret != SQL_SUCCESS_WITH_INFO)
    error (.conn.GetDiagRec());
```

If the connection fails, the return value will be something other than SQL_SUCCESS or SQL_SUCCESS_WITH_INFO. You can query the database for details of the failure by calling the GetDiagRec method of the ODBCConnection.

**Creating a Gamma Class from a Database Table**

The tutorial code connects to an existing database table. If you would like the script create a table for you, please refer to the section called “Creating a Database Table” below.

ODBC tables are mapped as classes into Gamma. This means that most interactions with the database will be through convenient method calls rather than SQL queries. Gamma is able to automatically determine the structure of a table in your database and to create a class from it:

```cpp
tableclass = .conn.ClassFromTable (#ClassName, nil, "table_name");
```
This statement will look up the table named `table_name` in your database, and create a class whose name is `ClassName` from it. It will also return the class definition into the variable `tableclass`.

Gamma attempts to determine the primary key field from your table. Some databases, such as Microsoft Access, do not provide a facility to do this. In that case, you need to assign the primary key for the table:

```csharp
.conn.SetPrimaryKey (tableclass, "id");
```

The resulting class will have a number of special data members, like `__table`, the name of the table, as well as a data member for each column of the database table.

### Querying Rows from the Database

Once you have mapped your database table to a Gamma class, you can query the database by constructing a `select` SQL call:

```csharp
allrows = .conn.QueryToClass (tableclass,
    string ("select * from ",
    tableclass.__table));
```

This statement will query all of the rows in the table attached to `tableclass`, and return them as an array in `allrows`. Each element of the array will be a class instance.

### Inserting Rows into a Database

To insert a row, first create a member of the class associated with the table, and then use `ODBCConnection.Insert` to insert it:

```csharp
local   timestamp;
local   record;

record = new (tableclass);
timestamp = localtime (clock());
timestamp = format ("%d-%02d-%02d %02d:%02d:%02d",
    timestamp.year + 1900, timestamp.mon + 1,
    timestamp.mday, timestamp.hour, timestamp.min,
    timestamp.sec);
record.ptname = "point name";
record.ptvalue = point_value;
record.pttimestamp = timestamp;
record.ptquality = point_quality;
.conn.Insert (record);
```

This code creates a record to be inserted, assigns a value to each column in the record, and then inserts it into the database. Gamma will attempt to convert the values in each column to the type required by the database. If an error occurs, the `Insert` method will throw an error that can be handled by a `try/catch` block.
You can also insert rows by constructing your own SQL statement and submitting it using the `ODBCConnection.QueryAndStore` method.

### Updating Existing Rows in a Database

To update an existing row in a database, you must know the primary key for the row you wish to update. Normally you find this key by performing a query on the database:

```lua
local   result = .conn.QueryToClass (tableclass, string
("select * from ",
tableclass.__table,
" where name = ",
"the_point_name", ","
));
local   record = result[0];
```

The primary key column of the record will identify the row in the database. You can now modify the record:

```lua
record.ptvalue = new_point_value;
```

and update the record in the database:

```lua
.conn.Update (record);
```

If an error occurs, the `Update` method will throw an error that can be handled by a `try/catch` block.

### Creating a Database Table

In most cases, you will create the database table using your database management software. This gives you a more convenient interface to the creation process. Different databases use different syntax to create a table. However, if you need to create a table within your script, there are two options:

1. Call the `ODBCConnection.CreateTable` method.
2. Call the `ODBCConnection.QueryAndStore` method.

The `CreateTable` method helps to construct the query, but it still requires you to understand the SQL syntax of table creation for your database.

```lua
.conn.CreateTable ("table name",
"id int PRIMARY KEY IDENTITY",
"ptname VARCHAR(20) NOT NULL",
"ptvalue DOUBLE NOT NULL",
"pttimestamp DATETIME NOT NULL",
"ptquality VARCHAR(20) NOT NULL"
);
```

The `CreateTable` arguments consist of the table name followed by any number of definitions for the columns in the table. The column definitions depend on the database be-
ing used. In particular, the primary key field (in this example, \texttt{id}) is very different from one database to another. The primary key must be integer, unique and auto-incrementing. In the case of Microsoft Access, you must issue an additional query to create a primary key:

```csharp
.conn CreateTable("table_name",
                 "id COUNTER",
                 "ptname VARCHAR(20) NOT NULL",
                 "ptvalue DOUBLE NOT NULL",
                 "pttimestamp DATETIME NOT NULL",
                 "ptquality VARCHAR(20) NOT NULL"
               );

.conn.QueryAndStore("create unique index p_id on table_name (id) with primary disallow null");
```

The alternative to using \texttt{ODBCConnection.CreateTable} is to construct your own complete SQL statement and submit it to the database using \texttt{ODBCConnection.QueryAndStore}:

```csharp
.conn.QueryAndStore("create table table_name
                     (id COUNTER, ptname VARCHAR(20) NOT NULL,
                      ptvalue DOUBLE NOT NULL,
                      pttimestamp DATETIME NOT NULL,
                      ptquality VARCHAR(20) NOT NULL"");

.conn.QueryAndStore("create unique index p_id on table_name (id) with primary disallow null");
```

If any call to \texttt{CreateTable} or \texttt{QueryAndStore} generates an error, then the error will be thrown. You can catch the error using a \texttt{try/catch} block within your script.

### Set up Event Handlers

The Gamma application will normally define functions to be called when an event occurs. In ODBC applications, this is most commonly to write live data into the relational database. Another word for an event is a "trigger". It is up to you to decide the best trigger for writing data into the database.

An event handler will look something like this:

```csharp
.OnChange (#$DataSim:Sine, '(@self).AddRecord (this, value));
```

This statement says that when the DataHub point \texttt{DataSim:Sine} changes, we want to call our application's \texttt{AddRecord} method with two arguments. The special variables \texttt{this} and \texttt{value} are always defined to be the point name and the point value respectively when running an \texttt{OnChange} handler. We might define our \texttt{AddRecord} method like this:

```csharp
method MyODBCScript.AddRecord (pointname, newvalue)
{
    local record = new (tableclass);
    local timestamp;
```
timestamp = localtime (PointGetUnixTime(pointname));
timestamp = format ("%d-%02d-%02d %02d:%02d:%02d",
    timestamp.year + 1900, timestamp.mon + 1,
    timestamp.mday, timestamp.hour, timestamp.min,
    timestamp.sec);
record.ptname = string (pointname);
record.ptvalue = number (newvalue);
record.pttimestamp = timestamp;
record.ptquality =
    GetQualityName(PointMetadata(pointname).ptquality);
    .conn.Insert (record);
record;
}

If you would like to add a record at a specific interval rather than on a value change, you could create a timer:

timerid = every (0.5, `(@self).AddRecord (#$DataSim:Sine, $DataSim:Sine));

This statement will add a new record to the database every ½ second. In this case, we need to specify the name of the point, $DataSim:Sine, as part of the AddRecord call because the special variables this and value are not defined during a timer call. The first reference to $DataSim:Sine is protected from evaluation during the call using the # modifier, so its name is passed to the first argument when the expression is evaluated (when the timer occurs). The second reference is not protected from evaluation, so its value is passed to the second argument.

You need to keep track of outstanding timers so that you can clean them up during the destructor:

method MyODBCScript.destructor ()
{
    cancel (timerid);
}

**Shut Down**

Normally an Application is not started and stopped repeatedly. The application is normally started when the DataHub starts, and continues until the DataHub is stopped.

To shut down your application, you need to destroy the application instance. This will cause the destructor to be called. Your clean-up code should be located in the destructor. The Application class automatically cleans up event handlers created using OnChange. You can detach from the ODBC database in your destructor:

method MyODBCScript.destructor ()
If you do not disconnect from the database in the destructor, Gamma will disconnect from the database when its garbage collector destroys the ODBCConnection object.

You can destroy the application instance by calling the destroy function. This can be triggered by a data point change or some other method. For example:

```csharp
.OnChange (#$default:kill_application, `destroy(@self));
```

This statement would wait for a change to the DataHub point named default:kill_application and then destroy the application. The application could then be restarted from the Scripting option in the DataHub Properties window.
Multi-Threaded ODBC Interface

The DataHub includes a scripting interface to multi-threaded database access. This comes in addition to the existing single-threaded ODBC implementation. A multi-threaded interface is substantially different from a single-threaded interface, and offers a number of advantages:

1. **Non-blocking** A single-threaded interface can block when attempting to communicate with a database that is no longer available. While blocked, all other Gamma scripts are halted until the database responds or the connection times out. This has a particularly substantial impact when attempting to communicate with more than one database at a time, since blocking on one database will also halt communication to the other database.

   With a multi-threaded interface, a separate thread is spawned to communicate with each database. If a communication problem occurs, only the database thread blocks. All other Gamma scripts continue to run normally.

2. **Store-and-forward** A single-threaded interface must be willing to drop data when the database becomes temporarily too busy to handle incoming data, or when the database is disconnected for some reason, such as a network failure.

   A multi-threaded interface can store data to disk when the database is unresponsive or unavailable. When the database recovers, the stored data can be transmitted to the database. Some multi-threaded implementations do not preserve operation order. Gamma's implementation guarantees that every operation will be performed in the same order that it was originally attempted.

3. **Time-delayed write** The multi-threaded implementation can store data to disk, to be written to the database at a later time or in batches as defined intervals. This is particularly useful if bandwidth is limited at certain times of day, or where the database connection is expensive to maintain at all times. For example, a database connection using satellite Internet could be charged based on connection time. It is much more efficient to store data temporarily and then connect at a predefined interval to update the database.

**How-To**

To ensure that the primary script thread does not block when there is a communication problem with the database, all communication to the database must be asynchronous. This means that a database command will normally not produce an immediate result. The result will be delivered at some time in future by triggering a "callback" function in the script. This is a different way of thinking about a script, and takes a little getting used to. Gamma provides a number of functions to make this as simple as possible.

**Create an ODBCThread Instance**

All communication with the database is performed through an instance of the class `ODBCThread`. For example:
mysql = new ODBCThread();
flags = STORE_AND_FORWARD;
cachefile = "C:/temp/mysql.cache";
mysql.Configure("myDSN", "username", "password", flags, cachefile);

This code is sufficient to create the database connection. At this point, the database is not connected, and the thread that handles the database has not yet been started.

Attach Event Callbacks

The next step is to attach the callback functions that will alert the script to various events from the database thread. Whenever a callback function is executed, a special variable called SQLResult will be defined for the duration of the callback. This contains information about the result of the SQL command, a description of the command and error codes. This information is stored in an instance of the class ODBCThreadResult.

A callback is attached by simply assigning the code to run to the callback member of the ODBCThread object. For example, to print information when the connection fails, you could do this:

```lua
method MyApp.onConnectionFail()
{
    princ ("Connection closed: ", SQLResult.Description, "\n");
}
```

and then in the application constructor, do this:

```javascript
thread.OnConnectionFailed = `(@self).onConnectFail();
```

The following callbacks are defined:

- **OnConnectionSucceeded**
  Called when the connection to the database transitions from not connected to connected.

- **OnConnectionFailed**
  Called when the connection to the database transitions from connected to not connected, or when a connection attempt fails. The Description member of the SQLResult contains information about the reason for the event.

- **OnExecuteStored**
  Called when a stored transaction is successfully forwarded to the database. The SQLResult contains information about the transaction.

- **OnFileSystemError**
  Called when a file system error occurs when reading or writing to the disk. The Description member for the SQLResult has details about the error.

- **OnODBCError**
  Called when an ODBC error occurs that cannot be reported as part of the result set from a successful transaction. Typically this would be a failure generated by an at-
tempt to execute a stored transaction.

**Configure Startup Actions**

One of the most difficult concepts of using an asynchronous interface is the idea that a sequence of steps must be performed in order during startup, even though the script code cannot be executed sequentially. At each step, there could be an asynchronous request/event pair that breaks up the sequence into disjoint code fragments.

The `ODBCThread` class defines a startup state machine that helps to sequence these steps. As a script developer, you define the steps that will be performed by creating a series of initialization steps, or "stages" that will be performed after a successful database connection. These stages will not automatically be executed. It is up to you to initiate them. This allows you to choose whether to execute these stages on each connection, or only on the first connection, or to start executing the stages at a stage other than the first.

To create initialization stages, make repeated calls to the method "addInitStage":

```csharp
AddInitStage(sqlCommand, onSuccess, onFailure);
```

The `sqlCommand` argument contains either a string containing valid SQL, or inline code to be executed. If the `sqlCommand` is a string, it is passed to the database thread for execution. If it is inline code, it will be executed immediately.

The `onSuccess` and `onFailure` arguments are inline code that will be executed when the `sqlCommand` has returned a result. Only one of these two will be executed for each execution of `sqlCommand`. The return value from `onSuccess` is ignored. If the return value from `onFailure` is `nil`, then the initialization sequence is aborted. It is sufficient to supply `nil` to `onFailure` to abort the sequence on any error, and to supply `t` to `onFailure` to continue the sequence if there is an error.

For example, the following sequence could be used:

```csharp
// Drop the existing table.  If we get an error, continue
thread.AddInitStage("drop table mytable", nil, t);

// Create a new table.  Once the table is created, set up an
// auto-incrementing primary key.  If either the table creation or
// the key definition fails, abort.
thread.AddInitStage("create table mytable ( myid counter,
                 myvalue number,
                 tstamp number )",

                 nil, nil);
thread.AddInitStage("create unique index p_myid on mytable (myid)
                      with primary disallow null", nil, nil);

// Make a call to getTableInfo to look up all tables in the
// database.  Since getTableInfo is asynchronous, we must use its
// callback to resume the initialization sequence by calling the
```
// special method "cbInitStage ()" in the callback, and then
// letting the initialization sequencer call the real callback
// mapTable is an example of user-created code.
thread.AddInitStage(`(@thread).getTableInfo("", "", "", "TABLE",
    `(@thread).cbInitStage()),
    `(@self).mapTable(@thread, "mytable",
        SQLTables), nil);

// We happen to know that the database we use does not provide
// primary key information so we have to set the primary key
// manually after the table has been mapped to class.
thread.AddInitStage(nil, `(@self).SetClassKey((@self).tableclass,
    #myid),
    nil);

// Finally, we have passed through all the initialization steps,
// so we begin storing data. beginDataStream in this example is
// user-generated code, which might set up timers or data event
// handlers that cause the script to write data to the database.
thread.AddInitStage(nil, `(@self).beginDataStream(@thread), nil);

Notice that the final two stages do not define the sqlCommand at all. This will cause their
success code to run immediately and then to move on to the next initialization stage. If an
initialization stage has no sqlCommand, it cannot fail.

At this point, the initialization stages are defined but have not run. They will not be run un-
til the script calls:

thread.beginAsyncInit();

Commonly, you only want to run the initialization once, so you might handle it in the On-
ConnectionSuccess handler like this:

method MyApp.onConnect (thread)
{
    prin "Connection succeeded\n";
    if (thread.is_first_connect)
    {
        thread.beginAsyncInit();
    }
}

Start the Database Thread

Once you have defined the parameters for the database connection, defined the callback
handlers and defined the initialization code, the connection is completely defined, but still
is not running. To start the thread and begin connecting, call the Start method:

thread.Start();
At this point the thread is started and begins trying to connect to the database. As the thread runs it may call your callback functions in any order to indicate successes, failures, and errors.

**Store and Forward**

*Store and Forward* is a term used to describe a database connection where the data is stored locally to disk and then later forwarded to the database. The *ODBCThread* object performs an advanced form of store and forward that does more than simply store data for later delivery.

For any SQL statement given to the *ODBCThread.ExecDirect* method, you can optionally specify that this particular statement should be stored for later forwarding. Normally these will be *INSERT* or *UPDATE* statements, but they could be an SQL statement that must be executed at the database, such as stored procedures or *ALTER* statements. The *ODBCThread* guarantees that all storable SQL commands will be executed in the order in which they are specified, even if they are first stored to file.

The *ODBCThread* object uses a sophisticated store and forward technique that only writes to disk if the database is not connected, or has been paused. If the database is available, the commands will be transmitted directly to the database. This means that there is no penalty to using store and forward during normal operation.

*ODBCThread* also maintains an in-memory queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. You can modify the depth of this queue to reduce the chance of involving the disk during busy periods. The queue depth defaults to 100 messages, but it can be modified by setting the *MaxTransactions* member of the *ODBCThread*. For example:

```plaintext
thread.MaxTransactions = 200;
```

For the thread to perform store and forward, the flag *STORE_AND_FORWARD* must be specified when initially configuring the thread, and the flag *STORE_AND_FORWARD* must also be specified for any call to *ExecDirect* that should be a candidate for store and forward treatment.

**Time Delayed Writes**

Time delayed writing is used to avoid maintaining a continuous connection to the database, or to make use of times of day where the network utilization is low. You can call the *Pause* and *Disconnect* methods at any time to cause the thread to pause output to the database, then disconnect. All further transactions will be written to the local disk cache until the database is reconnected. To resume writing to the database, the script just needs to call the *Resume* method. The database thread will automatically reconnect to the database when it can.

You can periodically test to see whether the disk cache has been transmitted by running a repeating timer that calls *CacheIsEmpty*. When *CacheIsEmpty* returns non-nil, the disk
cache has been consumed. At this point the script can once again Pause and Disconnect the thread.

Using this method, the script can transmit the cached data based on the time of day, a process event, or even input from an operator.

**Example**

```plaintext
/*
 * This script demonstrates the use of the threaded ODBC interface to insert data into a database based on a timer.
 */

require ("Application");
require ("ODBCThreadSupport");
require ("Time");
require ("Quality");

class ODBCThreadDemo Application
{
  DSN = "MySQLLocal";         // The DSN name to use for the database connection
  username = "root";          // The user name for connecting to the database
  password = "GY&*ik";        // The password for connecting to the database
  tablename = "andrew5";      // The name of the database table
  cachefile = "c:/tmp/testcache.txt";     // Base name for the disk cache file

  tableclass;
  thread;
}

/* This method will be called every time the connection is established to the database.
 * If there is something we only want to perform on the first connection, we can test is_first_connect to perform the code only once.
 */
method ODBCThreadDemo.onConnect()
{
  princ ("Connection succeeded\n");
  if (.thread.is_first_connect)
  {
    // Start the sequence defined by the AddInitStage calls in
```
// the constructor
d .thread.BeginAsyncInit();
}
}

/* If we get a connection attempt failure, or the connection fails
 * after having been connected, this method is called.
*/
method ODBCThreadDemo.onConnectFail()
{
    prin ("Connection closed: ", SQLResult.Description, "\n");
}

/* Map the table in the set of table definitions that matches the
 * name in .tablename into a Gamma class. This lets us easily
 * convert between class instances and rows in the table.
*/
method ODBCThreadDemo.mapTable(name, tabledefinitions)
{
    .tableclass = .thread.ClassFromTable(name, tabledefinitions);
}

/* Set up the timer or event handler functions to write to the table.
*/
method ODBCThreadDemo.startLogging()
{
    /* You can modify and/or add similar timers or event handlers
     * for each data point that you want to log. Please refer to
     * the "Methods and Functions from Application.g" section of
     * the documentaton for more details about the timer and event
     * handler funtions.
     */
    // Log a new row of data every 3 seconds.
    .TimerEvery(3, `(@self).writeData(#$DataSim:Sine));

    // Log a new row of data at 20 seconds past each minute
    // of each hour, etc.
    .TimerAt(nil, nil, nil, nil, nil, 20,
        `(@self).writeData(#$DataSim:Triangle));

    // Log a new row of data for the point DataSim:Square when
    // it changes.
    .OnChange(#$DataSim:Square, `(@self).writeData(this));

    // Log a new row of data for the point DataSim:Sine when
    // DataSim:Square changes.
OnChange($DataSim:Square, ~(@self).writeData($DataSim:Sine));

method ODBCThreadDemo.writeData(pointsymbol)
{
  local       row = new (.tableclass);
  local       pttime, ptltime;
  local       timestring;

  /* Generate a timestamp in database-independent format to the
   * millisecond. Many databases strip the milliseconds from a
   * timestamp, but it is harmless to provide them in case the
   * database can store them.
   */
  pttime = WindowsTimeToUnixTime(PointMetadata(pointsymbol).timestamp);
  ptltime = localtime(pttime);
  timestring = format("{ts '%04d-%02d-%02d %02d:%02d:%02d.%03d'}",
                      ptltime.year+1900, ptltime.mon+1,
                      ptltime.mday, ptltime.hour,
                      ptltime.min, ptltime.sec,
                      (pttime % 1) * 1000);

  /* Fill the row. Since we mapped the table into a Gamma class,
   * we can access the columns in the row as member variables of
   * the mapped class.
   */
  row.ptname = string(pointsymbol);
  row.ptvalue = eval(pointsymbol);
  row.pttime = timestring;

  /* Perform the insertion. In this case we are providing no
   * callback on completion.
   */
  .thread.Insert(row, nil);
}

/* Write the 'main line' of the program here. */
method ODBCThreadDemo.constructor ()
{
  // Create and configure the database connection object
  .thread = new ODBCThread();
  .thread.Configure(.DSN, .username, .password,
                    STORE_AND_FORWARD, .cachefile, 0);
/* Use this to delete the table on the first connection after
 * the script starts. BE CAREFUL - re-running the script will
 * start over and delete the table again.
 */

// .thread.AddInitStage(format("drop table %s", .tablename),
//                      nil, t);

/* Use this to create the table if it does not exist. Note: this
 * might not work for all databases. When in doubt, create the
 * table manually. The 't' in the onFail argument says to ignore
 * errors and continue with the next stage.
 */

// thread.AddInitStage(format
//    ("create table %s (ptid int
//      auto_increment primary key, ptname
//      varchar(64), ptvalue double, pttime
//      datetime )", .tablename), nil, t);

/* Query the table and map it to a class for each insertion.
 * We want to run an asynchronous event within the asynchronous
 * initialization stage, so to do that we specify the special
 * method cbInitStage as the callback function of our asynchronous
 * event (GetTableInfo). We deal with the return from the
 * GetTableInfo in the onSuccess argument of the init stage.
 */

.thread.AddInitStage(`(@.thread)
    .GetTableInfo("", ",
        (@.tablename),
        "TABLE,VIEW",
        `(@.thread).cbInitStage()),
    `(@self).mapTable(@.tablename,
        SQLTables),
    nil);

/* Do not start writing data to the table until we have
 * successfully created and mapped the table to a class. If we
 * wanted to start writing data immediately, then we would create
 * the table class beforehand instead of querying the database
 * for the table definition. Then, even if the database were
 * unavailable we could still cache to the local disk until the
 * database was ready.
 */

.thread.AddInitStage(nil, `(@self).startLogging(), nil);

/* Set up the callback functions for various events from the
 * database thread */
Multi-Threaded ODBC Interface

```ruby
/*
.thread.OnConnectionSucceeded = (@self).onConnect();
.thread.OnConnectionFailed = (@self).onConnectFail();
.thread.OnFileSystemError = princ("File System Error: ", SQLResult, "\n");
.thread.OnODBCError = princ("ODBC Error: ", SQLResult, "\n");
.thread.OnExecuteStored = nil;

/* Now that everything is configured, start the thread and begin
 * connecting. All of the logic now will be driven through the
 * onConnect callback and then through the init stages.
 */
.thread.Start();

/* Any code to be run when the program gets shut down. */
method ODBCThreadDemo.destructor ()
{
}

/* Start the program by instantiating the class. */
ApplicationSingleton (ODBCThreadDemo);
```
Classes
DATE_STRUCT

DATE_STRUCT — contains dates (y,m,d).

Synopsis

```cpp
class DATE_STRUCT
{
    day;
    month;
    year;
}
```

Description

This structure contains dates. For more information, please refer to C Data Types.
**ODBCColumn**

**ODBCColumn —**

**Synopsis**

```cpp
class ODBCColumn
{
    columnsize;
    datatype;
    decimaldigits;
    name;
    nullable;
}
```

**Description**

Not yet documented.
**ODBCConnection**

ODBCConnection — allocates a connection handle.

**Synopsis**

```latex
class ODBCConnection ODBCHandle
{
    __stmt;
    h;
    handle;
    type;
}
```

**Base Classes**

`ODBCHandle <-- ODBCConnection`

**Description**

This class allocates a **connection handle**. It corresponds to using the value `SQL_HANDLE_DBC` for the `HandleType` of the `SQLAllocHandle` function.

**Class Members**

These functions are identical to the corresponding C or C++ functions, as noted.

- `addColumn (tablename, column, type, default_val, allow_null)` corresponds to `addColumn`.
- `AllocDescriptor ()` corresponds to `AllocDescriptor`.
- `AllocStatement ()` corresponds to `SQLAllocStmt`.
- `ClassFromColumns (symclassname, superclass, columns)` corresponds to `ClassFromColumns`.
- `ClassFromTable (symclassname, superclass, tablename)` corresponds to `ClassFromTable`.
- `ClassesFromTables (superclass, verbose?=nil)` corresponds to `ClassesFromTables`.
- `Connect (ServerName, UserName, Authentication)` corresponds to `SQLConnect`.
CreateTable (tablename, columns...) corresponds to CreateTable.

Delete (row) corresponds to Delete.

Disconnect () corresponds to SQLDisconnect.

DropTable (tablename) corresponds to DropTable.

EndTran (CompletionType) corresponds to SQLEndTran.

Error () corresponds to SQLError.

GetOneColumn (query_string) corresponds to GetOneColumn.

GetOneValue (query_string) corresponds to GetOneValue.

Insert (row) corresponds to Insert.

MakeClass (symclassname, superclass, ivars, tablename, primary_key) corresponds to MakeClass.

MapClassFromResponse (klass, response) corresponds to MapClassFromResponse.

Query (query_string) corresponds to Query.

QueryAndStore (query_string) corresponds to QueryAndStore.

QueryToClass (klass, query_string) corresponds to QueryToClass.

QueryToTempClass (superclass, query) corresponds to QueryToTempClass.

ReQuery (row) corresponds to ReQuery.

Statement () corresponds to Statement.

StoreResult () corresponds to StoreResult.

Update (row) corresponds to Update.
Classes

(The following functions are inherited from: ODBCHandle)

GetDiagRec ()
    corresponds to SQLGetDiagRec.
ODBCDescriptor

ODBCDescriptor — allocates a descriptor handle.

Synopsis

class ODBCDescriptor ODBCHandle
{
    h;
    handle;
    type;
}

Base Classes

ODBCHandle <-- ODBCDescriptor

Description

This class allocates a descriptor handle. It corresponds to using the value SQLHANDLE_DESC for the HandleType of the SQLAllocHandle function.
**ODBCEnvironment**

ODBCEnvironment — allocates an environment handle.

**Synopsis**

```cpp
class ODBCEnvironment ODBCHandle
{
  h;
  handle;
  type;
}
```

**Base Classes**

ODBCHandle < ODBCEnvironment

**Description**

This class allocates an environment handle. It corresponds to using the value SQL_HAN-
DLE_ENV for the HandleType of the SQLAllocHandle function.

**Class Members**

These functions are identical to the corresponding C or C++ functions, as noted.

- AllocConnection ()
  - corresponds to SQLAllocConnect.

(The following functions are inherited from: ODBCHandle)

- GetDiagRec ()
  - corresponds to SQLGetDiagRec.
ODBCHandle

- **ODBCHandle** — a parent class for connections, descriptors, environments, and statements.

**Synopsis**

```cpp
class ODBCHandle
{
  handle;
  type;
}
```

**Description**

This class is a parent class for ODBCConnection, ODBCDescriptor, ODBCEnvironment, and ODBCStatement handles. Together, these classes provide the functionality of SQLAllocHandle.

**Class Members**

- These functions are identical to the corresponding C or C++ functions, as noted.

- GetDiagRec ()
  - corresponds to SQLGetDiagRec.
**ODBCResult**

**Synopsis**

```cpp
class ODBCResult
{
    columns;
    rows;
}
```

**Description**

Not yet documented.

**Class Members**

These functions are identical to the corresponding C or C++ functions, as noted.

`ColumnIndex (column_name)`

 corresponds to `ColumnIndex`. 
**ODBCStatement**

ODBCStatement — allocates a statement handle.

**Synopsis**

```cpp
class ODBCStatement ODBCHandle
{
    h;
    handle;
    type;
}
```

**Base Classes**

```
ODBCHandle <-- ODBCStatement
```

**Description**

This class allocates a statement handle. It corresponds to using the value `SQL_HANDLE_STMT` for the `HandleType` of the `SQLAllocHandle` function.

**Class Members**

These functions are identical to the corresponding C or C++ functions, as noted.

- **Cancel ()**
  - corresponds to `SQLCancel`.
- **CloseCursor ()**
  - corresponds to `SQLCloseCursor`.
- **Columns (CatalogName, SchemaName, TableName, ColumnName)**
  - corresponds to `SQLColumns`.
- **ExecDirect (StatementText)**
  - corresponds to `SQLExecDirect`.
- **Execute ()**
  - corresponds to `SQLExecute`.
- **Fetch ()**
  - corresponds to `SQLFetch`.
- **FetchScroll (FetchOrientation, FetchOffset)**
  - corresponds to `SQLFetchScroll`.
- **FreeStmt (Option)**
  - corresponds to `SQLFreeStmt`.
GetResultData ()
  corresponds to SQLGetData.

Prepare (StatementText)
  corresponds to SQLPrepare.

PrimaryKeys (CatalogName, SchemaName, TableName)
  corresponds to SQLPrimaryKeys.

RowCount (StatementText)
  corresponds to SQLRowCount.

Tables (CatalogName, SchemaName, TableName, TableType)
  corresponds to SQLTables.

(The following functions are inherited from: ODBCHandle)

GetDiagRec ()
  corresponds to SQLGetDiagRec.
ODBCThread

ODBCThread — configures the multi-threaded ODBC interface.

Synopsis

class ODBCThread
{
    Commands; // Internal variable
    Callbacks; // Internal variable
    OnExecuteStored; // callback
    OnConnectionSucceeded; // callback
    OnConnectionFailed; // callback
    OnFileSystemError; // callback
    OnODBCError; // callback
    NCached; // Internal variable
    NUncached; // Internal variable
    NStoredPrimary; // Number of transactions written to level 1 cache
    NStoredSecondary; // Number of transactions written to level 2 cache
    NForwarded; // Number of transactions read from cache
    NStoreFailPrimary; // Number of failures while writing to level 1 cache
    NStoreFailSecondary; // Number of failures while writing to level 1 cache
    NForwardFail; // Number of failures while writing to the database from cache
    NCommands; // Number of commands ever queued to thread
    NResults; // Number of commands results ever returned from thread
    NRejected; // Number of commands that were rejected without queueing
    ReconnectSecs; // Number of seconds to wait between database reconnection attempts
    ForwardDelay; // Delay in milliseconds between transactions written from cache to database
    MaxTransactions; // Maximum number of transactions to hold on queue
}
Description

As of this writing, the ForwardDelay member is not implemented.

The script developer should only modify the following members:

- OnExecuteStored
- OnConnectionSucceeded
- OnConnectionFailed
- OnFileSystemError
- OnODBCError
- ReconnectSecs
- ForwardDelay
- MaxTransactions

Methods

- ODBCThread.CacheIsEmpty()
  returns non-nil if both the level 1 and level 2 cache files are empty.

- ODBCThread.Columns (catalog, schema, tablename, columnname, callback)
  queries the database table for its column definitions. The catalog, schema, table-
  name and columnname are all strings. Some of these may accept wildcard patterns de-
  pending on the database being used. When the call completes, the callback code will
  be executed. The result is available in the SQLResult for the duration of the callback.

- ODBCThread.Configure (DSN, username, password, flags, storagefile, maxfilesize)
  sets the initial configuration for the database connection. The flags parameter can
  be either 0 or STORE_AND_FORWARD. If STORE_AND_FORWARD is not specified then no
  command on this connection will be stored, even if the individual command specifies
  the STORE_AND_FORWARD flag. The storagefile parameter specifies the name of a
  file to store the level 1 cache information for store and forward operation. The level 2
  cache file name will be created from this file name.
  
  The maxfilesize specifies the maximum number of bytes that a cache file can grow
  to. There can in fact be 3 cache files, each of this size, at any one time. The maxfile-
  size may be exceeded by the length of a single transaction for any file. If you set
  maxfilesize to 0, then 2.1 GB will be used. If the size exceeds this amount then it
  will be truncated to 2.1 GB. You may wish to intentionally limit the file size to a lower
  number. In a system where the data rate is always too high for the database to han-
  dle, a smaller cache file size will cause the ODBCThread to go through periods where
  its data is discarded while the cache file is caught up. A smaller file will make this dis-
  card/catch-up cycle faster.
  
  Flags can be any combination of:
• **STORE_AND_FORWARD** - If this flag is not set, then no file storage will take place. Any transactions that cannot be written to the thread immediately will be rejected. Any transactions in the queue that cannot be delivered to the database will be discarded.

• **NO_STORE_TO_SECONDARY** - This tells the ODBCThread to only use the level 1 cache. If the queue between the script and the database thread becomes full, further transactions will be rejected until there is space in the queue. However, any transactions in the queue that cannot be delivered to the database will be stored in level 1 cache.

• **STORE_ALWAYS** - This flag tells the ODBCThread to always store a transaction to disk before sending it to the database. This will normally cause the thread to read its queue faster, and write the transactions to disk more frequently. In case of a system crash, those transactions are more likely to be recoverable when the script re-starts. This option imposes a speed penalty if the disk is slow. On systems with fast disks, this penalty is normally minimal.

• **ALLOW_CACHE_RESTART** - In case of a system or application crash, the ODBCThread will resume reading any disk files at the point where it left off when the application restarts. This flag tells the ODBCThread not to track its position in the disk file, and to restart at the beginning of the file during a crash recovery. This improves speed on systems with a slow disk, but means that some transactions may be sent to the database more than once. This should only be used if disk access is slow.

• **NO_FLUSH_ON_WRITE** - The ODBCThread normally tries to update files as soon as possible after a write to disk. This is not efficient, but it improves the chance that more transactions will be recoverable in the case of a system or application crash. Specifying this flag will cause the ODBCThread to store data in memory longer and write to disk in larger blocks. This may improve performance for systems with a slow file system, but it increases the chance that transactions will be lost if the system crashes or shuts down.

We recommend using either:

```
STORE_AND_FORWARD
```

or

```
STORE_AND_FORWARD | STORE_ALWAYS
```

unless the impact of disk access is unacceptably high on the system.

ODBCThread.Connect ()

forces a connection attempt, even if the thread connection timer has not expired.

ODBCThread.DataSources (type)

lists all data sources (DSNs). The type parameter can be one of:

• **SQL_FETCH_FIRST** - retrieve all DSNs.

• **SQL_FETCH_FIRST_USER** - retrieve only user DSNs.

• **SQL_FETCH_FIRST_SYSTEM** - retrieve only system DSNs.

The return value is an array of lists of the form: {"dsn_name", "dsn_driver"}
Classes

ODBCThread.Disconnect()
forces the thread to disconnect from the database. If the thread is not paused then it will attempt to reconnect to the database on the next reconnect timer cycle.

ODBCThread.ExecDirect(flags, sql, callback)
executes an SQL statement on the database. Flags can be either 0 or STORE_AND_FORWARD. If the command cannot be executed immediately, and STORE_AND_FORWARD is set, and STORE_AND_FORWARD is also set on the thread, then the command will be stored to file and executed later. The SQL statement is a string. When the statement is executed, the callback will be called. The result is available in the SQLResult for the duration of the callback.

ODBCThread.GetFlags()
retrieves the flags set by the .Configure method.

ODBCThread.GetMessageCount()
retrieves the number of messages currently queued to the database thread.

ODBCThread.GetResultCount()
retrieves the number of results currently queued from the database thread to the script thread.

ODBCThread.Insert(row, callback)
performs a database INSERT given an instance of a class that has been mapped to a column set in the database. The row must be an instance of a class returned from .ClassFromResultSet, .ClassFromTable, or .ClassFromThreadResult. When the insertion is complete, the callback is executed.

ODBCThread.QueueIsFull()
returns non-nil if the message queue is full.

ODBCThread.IsPaused()
returns non-nil if the thread is paused. See the information in ODBCThread.Pause().

ODBCThread.NoOp(callback)
sends a message to the database thread, and do nothing. When the message has arrived at the database thread, the method returns and runs the callback. This is a mechanism to synchronize execution in the script with actions that are queued on the database thread.

ODBCThread.Pause()
pauses the thread. A paused thread will continue to store data to disk to be forwarded later, but it will not perform transactions on the database. If the database is disconnected and paused, the thread will not attempt to reconnect until the thread is resumed.

ODBCThread.PrimaryKeys(catalog, schema, tablename, callback)
queries the database for the primary keys for the given catalog, schema and tablename. The result is available in the SQLResult when the callback is executed.

ODBCThread.QuoteConversion(head, tail, character, replacement)
is used internally.
ODBCThread.Resume ()
    resumes a thread that has been previously paused by .Pause().

ODBCThread.SlowInsert (row, callback)
    is an alternate (slower) method to insert data. It acts the same as the .Insert method, except that it recomputes the SQL statement on each insert. The .Insert method computes the SQL statement ahead of time.

ODBCThread.SlowUpdate (row, callback)
    is an alternate (slower) method to update data. It acts the same as the .Update method, except that it recomputes the SQL statement on each update. The .Update method computes the SQL statement ahead of time.

ODBCThread.Start ()
    starts the thread and begins attempting to connect.

ODBCThread.Stop ()
    closes the connection to the database and stops the thread.

ODBCThread.Tables (catalog, schema, tablename, tabletype, callback)
    queries the database for all tables matching the catalog, schema, tablename and tabletype. It calls the callback when the transaction completes. Specifying an empty string ("") for any argument indicates no preference. The tabletype must be one of "TABLE", "VIEW" or "TABLE, VIEW". The result of this call is available in ODBCResult.

ODBCThread.Update (row, callback)
    performs a database UPDATE given an instance of a class that has been mapped to a column set in the database. The row must be an instance of a class returned from .ClassFromResultSet, .ClassFromTable or .ClassFromThreadResult. When the update is complete, the callback is executed. The result of this call is available in ODBCResult.

ODBCThread.ValueString (value)
    is used internally.

ODBCThread.AddInitStage (sqlString, onSuccess, onFailure)
    adds an initialization stage to the sequential set of steps to be executed as part of the initialization after the database connections is made. See the section Configure Start-up Actions above. The return value from this function is an index that can be given to .BeginAsyncInit.

ODBCThread.BeginAsyncInit (stage?=0)
    starts executing the initialization stages in the order in which they were specified. If the stage argument is non-zero, being executing from that index in set. This index is provided as the return value from .AddInitStage.

ODBCThread.cbInitStage ()
    is a provided callback that can be used to trigger the next initialization stage in the initialization sequence. If the stage specifies user-defined code instead of a string for the sqlString argument of .AddInitStage, that code must at some point call
.cbInitStage in order for the sequence to continue.

ODBCThread.ClassFromResultSet (columnresult, keyresult, superclass? =nil, symclassname?=#UnboundODBCThreadTableClass)
creates a class from the table defined in the given result sets. The columnresult is the column definition for the table, and the keyresult is the result from calling .PrimaryKeys on the table, or the result from querying the table through the .Tables method. If the superclass is non-nil, the class will be derived from superclass, otherwise it will have no parent class. If symclassname is provided, that class name will be used instead of the default UnboundODBCThreadTableClass. The class produced by this call maps each column in the columnresult to a member variable in the class. In addition, information about the primary key and the source table is held in the class. Instances of this class are suitable for use with the .Insert method. If the table has a primary key, then instances of this class can also be used in the .Update method.

ODBCThread.ClassFromTable (tablename, tables, superclass?=nil, symclassname?=#UnboundODBCThreadTableClass)
creates a class from the table named tablename from the table set defined in tables. The tables argument is the value of SQLTables from a call to the .GetTableInfo method. See the discussion in .ClassFromResultSet for more information.

ODBCThread.ClassFromThreadResult (threadresult, superclass?=nil, symclassname?=#UnboundODBCThreadTableClass)
creates a class from an ODBCThreadResult instance. This instance is usually obtained from a call to .GetTableInfo or .Columns. See the discussion in .ClassFromResultSet for more information.

ODBCThread.constructor ()
is the constructor for this class. Do not override the constructor for ODBCThread in your own code. Instead, derive a new class from ODBCThread and then define a constructor for your derived class.

ODBCThread.CreateClass (symclassname, superclass, ivars, tablename, primary_key)
is the low-level call made from .ClassFromResultSet, .ClassFromTable and .ClassFromThreadResult. It constructs the necessary code to define a class that maps its member variables to columns in a result set.

ODBCThread.EvalSafe (code)
is used internally.

ODBCThread.GetDataSources (direction?=SQL_FETCH_FIRST)
queries the ODBC subsystem for the names of all DSNs. The type parameter can be one of:

• SQL_FETCH_FIRST - retrieve all DSNs.
• SQL_FETCH_FIRST_USER - retrieve only user DSNs.
• SQL_FETCH_FIRST_SYSTEM - retrieve only system DSNs.
Classes

The return value is an array of lists of the form: ("dsn_name" . "dsn_driver")
This method is the only method that calls synchronously into the ODBC subsystem.
The result is available as the return value from this function. The ODBC definition
states that this call will be entirely satisfied by the driver manager, and so cannot block
on the database. The database does not need to be connected, and the database
thread does not have to be started for this method to succeed.

ODBCThread.GetInsertFormat (klass)
constructs the SQL statement that will be issued when the .Insert method is called.
If you change the definition of the table by calling .SetClassKey, then you must also

ODBCThread.GetTableInfo (catalog, schema, tablename, tabletype, callback)
produces a result that will be available in the special variable SQLTables for the du-
ration of the callback. SQLTables is an array of arrays. Each element of the result is
an array of two elements containing the table name and an instance of ODBCThread-
Result corresponding to a call to .Columns for that table.

ODBCThread.GetUpdateFormat (klass)
constructs the SQL statement that will be issued when the .Update method is called.
If you change the definition of the table by calling .SetClassKey, then you must also

ODBCThread(HandleColumnInfo (tablename, results, callback)
is used internally.

ODBCThread.HandleFinalInfo (results, callback)
is used internally.

ODBCThread.HandleTableInfo (results, callback)
is used internally.

ODBCThread.NextInitStage ()
is used internally.

ODBCThread.SetClassKey (klass, keysym, ignore_if_set?=t)
sets the primary key for the class specified by klass. The class is the result of a call to
.ClassFromResultSet, .ClassFromTable and .ClassFromThreadResult. Some
databases (MS-Access in particular) do not provide information about the primary
keys in a table. In order for .Update calls to work on this type of class, the .Set-
ClassKey method must be called to tell the table which column is its primary key.
**ODBCThreadResult**

**Synopsis**

```java
class ODBCThreadResult {
    Result;        // The SQL result set, or nil.
    KeyResult;     // The SQL result set describing the primary
                  // keys, for those commands that produce a
                  // primary key set.
    Diagnostic;    // The complete ODBC diagnostic set, if any.
    KeyDiagnostic; // The diagnostic set for the primary key query.
    Description;   // A description of the command or result.
    ReturnCode;    // A numeric SQLRESULT for an SQL command, or an
                  // "errno" return code for a file system error.
    AffectedRows;  // The number of affected rows for an SQL command,
                  // if available.
}
```

**Description**

The `Diagnostic` and `KeyDiagnostic` each consist of an array, where the elements of the array are themselves arrays:

- `element[0] = the database diagnostic message, as a string.```
- `element[1] = the ODBC diagnostic state code, as a string.```
- `element[2] = the ODBC native error code, as a number.```

There can be more than one diagnostic message returned by a single SQL call.

The `Result` and `KeyResult` contain the column and row definitions resulting from an SQL command. This definition is shared with the single-threaded ODBC implementation. The complete definition for `ODBCResult` is:

```java
class ODBCResult {
    columns;
    rows;
}
```

The `columns` member is an array of instances of the `ODBCColumn` class:

```java
class ODBCColumn {
    columnsize;     // A numeric column width.
```
datatype; // A number representing the ODBC data type.
decimaldigits; // The number of decimal digits, or 0.
name;        // The column name.
nullable;    // 0 if not nullable, 1 if nullable.
}

The rows member is an array of values, each of which corresponds to the column definition in the same position in the columns array.
SQLGUID

SQLGUID — holds ID strings.

Synopsis

class SQLGUID
{
}

Description

This structure is used to hold ID strings. For more information, please refer to C Interval Structure.
**SQL_DAY_SECOND_STRUCT**

*SQL_DAY_SECOND_STRUCT* — contains time data for *SQL_INTERVAL_STRUCT*.

**Synopsis**

```cpp
class SQL_DAY_SECOND_STRUCT
{
    day;
    fraction;
    hour;
    minute;
    second;
}
```

**Description**

This structure contains time data for *SQL_INTERVAL_STRUCT*. For more information, please refer to [C Interval Structure](#).
SQL_INTERVAL_STRUCT

SQL_INTERVAL_STRUCT — contains interval data for SQL queries.

Synopsis

class SQL_INTERVAL_STRUCT
{
    interval_sign;
    interval_type;
}

Description

This structure contains interval data for SQL queries. For more information, please refer to C Interval Structure.
**SQL_INTERVALSTRUCT_intval**

`SQL_INTERVALSTRUCT_intval` — contains year/month or day/second info for `SQL_INTERVAL_STRUCT`.

**Synopsis**

```cpp
class SQL_INTERVALSTRUCT_intval
{
    day_second;
    year_month;
}
```

**Description**

This structure contains year/month or day/second info for `SQL_INTERVAL_STRUCT`. For more information, please refer to [C Interval Structure](#).
SQL_NUMERIC_STRUCT

SQL_NUMERIC_STRUCT — specifies number precision and sign.

Synopsis

```cpp
class SQL_NUMERIC_STRUCT
{
    precision;
    sign;
}
```

Description

This structure specifies number precision and sign. For more information, please refer to C Data Types.
SQL_YEAR_MONTH_STRUCT

SQL_YEAR_MONTH_STRUCT — contains year and month data for SQL_INTERVAL_STRUCT.

Synopsis

class SQL_YEAR_MONTH_STRUCT
{
   month;
   year;
}

Description

This structure contains year and month data for SQL_INTERVAL_STRUCT. For more information, please refer to C Interval Structure.
**TIMESTAMP_STRUCT**

**Synopsis**

```c
class TIMESTAMP_STRUCT
{
    day;
    fraction;
    hour;
    minute;
    month;
    second;
    year;
}
```

**Description**

This structure contains timestamp data. For more information, please refer to C Data Types.
TIME_STRUCT

TIME_STRUCT — contains time data (h,m,s).

Synopsis

class TIME_STRUCT
{
    hour;
    minute;
    second;
}

Description

This structure contains time data. For more information, please refer to C Data Types.
Global Functions
**ODBC_AllocEnvironment**

ODBC_AllocEnvironment — **creates** an ODBCEnvironment.

**Synopsis**

```
ODBC_AllocEnvironment()
```

**Description**

This function is used to create an **ODBCEnvironment** class, the first step in creating an ODBC database. When you allocate the ODBC environment in your script, you can optionally specify the ODBC version that you want. To do this, give the version number (2 or 3) to `ODBC_AllocEnvironment`, like this: `ODBC_AllocEnvironment(2);`
**ODBC_ValueString**

**Synopsis**

```
ODBC_ValueString (value)
```

**Description**

Not yet documented.
Constants

ODBCVER
SQL_ACCESSIBLE_PROCEDURES
SQL_ACCESSIBLE_TABLES
SQL_ALL_TYPES
SQL_ALTER_TABLE
SQL_AM_CONNECTION
SQL_AM_NONE
SQL_AM_STATEMENT
SQL_API_SQLALLOCCONNECT
SQL_API_SQLALLOCCENV
SQL_API_SQLALLOCHANDLE
SQL_API_SQLALLOCSMT
SQL_API_SQLBINDCOL
SQL_API_SQLBINDPARAM
SQL_API_SQLCANCEL
SQL_API_SQLCLOSECURSOR
SQL_API_SQLCOLATTRIBUTE
SQL_API_SQLCOLUMNS
SQL_API_SQLCONNECT
SQL_API_SQLCOPYDESC
SQL_API_SQLDATASOURCES
SQL_API_SQLDESCREBCOL
SQL_API_SQLDISCONNECT
SQL_API_SQLENDTRAN
SQL_API_SQLERROR
SQL_API_SQLEXECDIRECT
SQL_API_SQLEXECUTE
SQL_API_SQLFETCH
SQL_API_SQLFETCHSCROLL
SQL_API_SQLFREECONNECT
SQL_API_SQLFREEENV
SQL_API_SQLFREEHANDLE
SQL_API_SQLFREESTMT
SQL_API_SQLGETCONNECTATTR
SQL_API_SQLGETCONNECTOPTION
SQL_API_SQLGETCURSORNAME
SQL_API_SQLGETDATA
SQL_API_SQLGETDESCFIELD
SQL_API_SQLGETDESCREC
SQL_API_SQLGETDIAGFIELD
SQL_API_SQLGETDIAGREC
SQL_API_SQLGETENVATTR
SQL_API_SQLGETFUNCTIONS
SQL_API_SQLGETINFO

SQL_DIAG_SERVER_NAME
SQL_DIAG_SQLSTATE
SQL_DIAG_SUBCLASS_ORIGIN
SQL_DIAG_UNKNOWN_STATEMENT
SQL_DIAG_UPDATE_WHERE
SQL_DOUBLE
SQL_DROP
SQL_ERROR
SQL_FALSE
SQL_FETCH_ABSOLUTE
SQL_FETCH_DIRECTION
SQL_FETCH_FIRST
SQL_FETCH_LAST
SQL_FETCH_NEXT
SQL_FETCH_PRIOR
SQL_FETCH_RELATIVE
SQL_FLOAT
SQL_GETDATA_EXTENSIONS
SQL_HANDLE_DBC
SQL_HANDLE_DESC
SQL_HANDLE_ENV
SQL_HANDLE_STMT
SQL_IC_LOWER
SQL_IC_MIXED
SQL_IC_SENSITIVE
SQL_IC_UPPER
SQL_IDENTIFIER_CASE
SQL_IDENTIFIER_QUOTE_CHAR
SQL_INDEX_ALL
SQL_INDEX_CLUSTERED
SQL_INDEX_HASHED
SQL_INDEX_OTHER
SQL_INDEX_UNIQUE
SQL_INSENSITIVE
SQL_INTEGER
SQL_INTEGRITY
SQL_INVALID_HANDLE
SQL_IS_DAY
SQL_IS_DAY_TO_HOUR
SQL_IS_DAY_TO_MINUTE
SQL_IS_DAY_TO_SECOND
SQL_IS_HOUR
SQL_IS_HOUR_TO_MINUTE
SQL_IS_HOUR_TO_SECOND
| SQL DIAG DROP TABLE                  | SQL TRANSACTION CAPABLE         |
| SQL DIAG DROP TRANSLATION           | SQL TRANSACTION ISOLATION OPTION |
| SQL DIAG DROP VIEW                  | SQL TRUE                       |
| SQL DIAG DYNAMIC DELETE_CURSOR     | SQL_TXN_CAPABLE                |
| SQL DIAG DYNAMIC FUNCTION          | SQL_TXN ISOLATION OPTION       |
| SQL DIAG DYNAMIC FUNCTION_CODE     | SQL_TYPE_DATE                  |
| SQL DIAG DYNAMIC UPDATE_CURSOR     | SQL_TYPE_TIME                  |
| SQL DIAG GRANT                     | SQL_TYPE_TIMESTAMP             |
| SQL DIAG INSERT                    | SQL_UNBIND                     |
| SQL DIAG MESSAGE_TEXT              | SQL_UNKNOWN_TYPE               |
| SQL DIAG NATIVE                    | SQL_UNNAMED                    |
| SQL DIAG NUMBER                    | SQL_UNSPECIFIED                |
| SQL DIAG RETURNCODE               | SQL_USER_NAME                  |
| SQL DIAG REVOKE                    | SQL_VARCHAR                    |
| SQL DIAG ROW_COUNT                 | SQL_XOPEN_CLI_YEAR             |
| SQL DIAG SELECT_CURSOR             |                               |
Cogent DataHub® Windows Scripting

Version 9.0
## Table of Contents

Introduction .......................................................................................................................... 1  
Overview ............................................................................................................................ 1  
Auto-Generation ................................................................................................................. 1  
Documentation ..................................................................................................................... 2  

Tutorials ............................................................................................................................... 3  
Making a Window .................................................................................................................. 3  
Displaying Data .................................................................................................................... 5  
Entering Data ......................................................................................................................... 9  
A complete demo program: WindowsExample.g ................................................................. 12  
Sample Code: ShowImage.g ................................................................................................. 25  
Sample Code: ListBoxExample.g ......................................................................................... 26  
Sample Code: TreeViewDemo.g ............................................................................................ 28  
Sample Code: Browse DataHub Points using TreeViewExample.g ..................................... 34  

Widgets .................................................................................................................................. 37  
G3StateButton ...................................................................................................................... 38  
GAnimateCtrl ....................................................................................................................... 39  
GBitmapButton ..................................................................................................................... 40  
GButton .................................................................................................................................. 41  
GCheckBox ............................................................................................................................ 43  
GCheckListViewCtrl ............................................................................................................. 44  
GColorDialog ....................................................................................................................... 45  
GComboBox .......................................................................................................................... 46  
GComboBoxEx ....................................................................................................................... 49  
GDateTimePickerCtrl ............................................................................................................ 51  
GDialog .................................................................................................................................. 52  
GDragListBox ....................................................................................................................... 53  
GEdit ...................................................................................................................................... 54  
GFileDialog .......................................................................................................................... 58  
GFlatScrollBar ..................................................................................................................... 60  
GFolderDialog ...................................................................................................................... 61  
GFontDialog ......................................................................................................................... 62  
GGroupBox ............................................................................................................................ 63  
GHeaderCtrl .......................................................................................................................... 64  
GHotKeyCtrl .......................................................................................................................... 66  
GHyperLink .......................................................................................................................... 67  
GIPAddressCtrl ..................................................................................................................... 67  
GLinkCtrl .................................................................................................................................. 70  
GLListBox .................................................................................................................................. 71  
GListViewCtrl ....................................................................................................................... 74  
GPagerCtrl ............................................................................................................................. 84  
GPageSetupDialog .................................................................................................................. 83  
GPrintDialog ........................................................................................................................ 86  
GProgressBarCtrl ................................................................................................................. 87  
GTreeCtrl ................................................................................................................................ 90  
G3StateButton ...................................................................................................................... 92  
GToolBar ................................................................................................................................ 93  
GTreeCtrl ................................................................................................................................ 94  
GComboBox .......................................................................................................................... 96  
GComboBoxEx ....................................................................................................................... 97  
GComboBox .......................................................................................................................... 98  
GComboBox .......................................................................................................................... 99  
GComboBox .......................................................................................................................... 100
GRadioButton ............................................................................................................... 89
GReBarCtrl ............................................................................................................. 90
GRichEditCtrl ......................................................................................................... 93
GScrollBar ............................................................................................................... 98
GStatic ..................................................................................................................... 100
GStatusBarCtrl .................................................................................................... 101
GTabCtrl ................................................................................................................ 103
GToolBarCtrl ......................................................................................................... 106
GToolTipCtrl .......................................................................................................... 111
GTrackBarCtrl ........................................................................................................ 114
GTreeViewCtrl ...................................................................................................... 117
GTreeViewCtrlEx .................................................................................................. 121
GUpDownCtrl .......................................................................................................... 122
GWindow ............................................................................................................... 124
GWindowBase ....................................................................................................... 125
Global Functions .................................................................................................. 134
Non-Widget Classes ............................................................................................... 141
Constants ................................................................................................................ 146
Introduction

Overview

The DataHub has built-in scripting capabilities based on Cogent's Gamma language, which among other things let you create Windows interfaces with DataHub scripts. This guide assumes a basic understanding of DataHub scripting, and provides the specific information you will need to create windows, buttons, frames, tabs, entry fields, and so on—all animated with live data.

DataHub Windows scripting combines two APIs:

- WTL-based widget classes
- Windows Platform SDK functions, structures and constants

Auto-Generation

The API is machine-generated, so some of the less common Windows functions are not supported. We plan to add those functions by hand on a case-by-case basis as they are identified as necessary. In all, there are 46 widget classes, 400+ common structures, 1700+ functions and 6000+ constants included. We do not plan to fully document it, but instead list the functions we support and rely on the reader to search the Microsoft documentation for details.

We have added some functions to make it convenient to set colors and fonts, and to anchor widgets for resizing support, but for the most part the functions we support are faithful to the Microsoft documentation.
Documentation

In addition to the API itself, about 95% of the reference sections of this documentation are automatically generated. In a few instances we have not been able to automatically extract all of the documentation from the code. The most notable example is when there is more than one calling sequence for the member function of a widget. Each of these functions has an argument list that looks like this: \((\text{args?...})\). Until we are able to extract the correct calling sequences and their syntaxes, you will have to search and read the Windows documentation and do a little trial and error to determine what the possibilities are.
Tutorials

The tutorials given here show how to create a window where you can display and interact with live data. These tutorials build on the tutorials in the DataHub Scripting manual and assume you have some knowledge of Gamma.

Making a Window

This first tutorial is repeated from the DataHub Scripting manual.

1. Open the Properties window, select the **Scripting** option, and click the **New** button to create a new script.

2. In the New Script File dialog, name the main class 'MyWindows' and select the **Windows** option. More details.

3. Add the file to your list of files, and load it now. Here's how. A new window will open:

4. Click the close icon in the top right corner. The window will close, and you will see this dialog box:

   ![Dialog Box]

The Code

The code that gets written to the `MyWindows.g` file is as follows:
/* All user scripts should derive from the base "Application" class */
require ("Application");

/* Get the Gamma library functions and methods for ODBC and/or
 * Windows programming. Uncomment either or both. */
require ("WindowsSupport");
//require ("ODBCSupport");

/* Applications share the execution thread and the global name
 * space, so we create a class that contains all of the functions
 * and variables for the application. This does two things:
 * 1) creates a private name space for the application, and
 * 2) allows you to re-load the application to create either
 *    a new unique instance or multiple instances without
 *    damaging an existing running instance.
 */
class MyWindows Application
{
  window;
}

/* Use methods to create functions outside the 'main line'. */
method MyWindows.samplemethod ()
{
}

/* Write the 'main line' of the program here. */
method MyWindows.constructor ()
{
  local rect = CreateRect (0, 0, 300, 300), txt;
  .window = new GWindow();

  .window.Create (0, rect, "Hello", WS_OVERLAPPEDWINDOW, 0);
  .window.CenterWindow();
  txt = .window.CreateControl (GStatic, 0, 0, 280, 22, "Hello world", SS_CENTER);
  txt.CenterWindow();
  .window.MessageHandler (WM_DESTROY, `(!destroyed_p(@self) ? destroy(@self) : nil));
  .window.ShowWindow (SW_SHOW);
}

/* Any code to be run when the program gets shut down. */
method MyWindows.destructor ()
{
// The WM_DESTROY message could come before or after this destructor depending
// on whether the application instance is destroyed or the window is closed
// first. We protect against the case where the window is closed first.
if (instance_p(.window) && .window.GetHwnd() != 0)
  .window.SendMessage (WM_CLOSE, 0, 0);
MessageBox(0, string ("Application: ", class_name(self), " completed."), "Done",
);   
/* Start the program by instantiating the class. If your
* constructor code does not create a persistent reference to
* the instance (self), then it will be destroyed by the
* garbage collector soon after creation. If you do not want
* this to happen, assign the instance to a global variable, or
* create a static data member in your class to which you assign
* 'self' during the construction process. ApplicationSingleton()
* does this for you automatically. */
ApplicationSingleton (MyWindows);

Displaying Data

This tutorial illustrates four ways to display live data in a window: as text, with a progress bar, with a custom progress bar, or in a list.

Output

![Widgets that can display live data]

Code

```javascript
require ("Application");
require ("WindowsSupport");
```
class MyDataDisplay Application
{
    window;
}

/* Adds items to a GListView widget. */
method MyDataDisplay.AddPoint (name, val, comment)
{
    local n = lv.GetItemCount();
    lv.InsertItem (n, name);
    lv.SetItemText (n, 1, val);
    lv.SetItemText (n, 2, comment);
}

/* Changes the 2nd value of a GListView item. */
method MyDataDisplay.ChangeVal (name, val)
{
    local finfo = new LVFINDINFO();
    finfo.flags = LVFI_STRING;
    finfo.psz = name;
    local item = lv.FindItem (finfo, -1);
    lv.SetItemText (item, 1, string(val));
}

/* Formats a label for convenience. */
function format_label (win, letterh, letterw, font, color)
{
    win.SetForeground (0, 0, color);
    win.SetFontEx (letterh, letterw, 0, 0, 0, 0, 0, 0, 0,
                   0, 0, 0, 0, font);
}

/* Updates a value on a GStatic widget. */
function update (win, point)
{
    local val, i;
    if (!undefined_p (val = eval(point)))
        point = string (point);
    if (((i = strchr (point, ':')) != -1)
        point = substr (point, i+1, -1);
    win.SetWindowText (format ("%s = %.4f", point, val));
}

/* Calculates the resize dimensions for a GStatic widget. */
function xrect (ptpos, low, high, pixels)
{  
  local ret = int (((ptpos - low)/(high - low)) * (pixels));
}

/* The 'main line' of the program. */
method MyDataDisplay.constructor ()
{
  local rect = CreateRect (0, 0, 300, 300), txt;
  local x = 10, y = 10, w = 370, h = 20;

  /* Create the window, size and position it. */
  .window = new GWindow();
  .window.Create (0, rect, "Data Display", WS_OVERLAPPEDWINDOW, 0);
  .window.MoveWindow(20, 200, 400, 345);

  /* Set the same background color for the window * 
   and its children. */
  .window.SetBackground(0, GetSysColor (COLOR_3DFACE), 0);
  .window.SetChildBackground(0, GetSysColor (COLOR_3DFACE), 0);

  /* Allow for the possibility that DataSim isn't running. 
   * Use the datahub_command function to send a cset command so 
   * that the point will get created if it doesn't already exist. */
  $DataSim:Sine := 0;
  datahub_command("(cset DataSim:Sine 0)");
  datahub_command("(cset DataSim:Ramp 0)");
  datahub_command("(cset DataSim:Triangle 0)");

  /* Set a frequency and update rate for DataSim that suits 
   * this demo. */
  datahub_command("(cset DataSim:Frequency 0.02)");
  datahub_command("(cset DataSim:UpdateFrequency 5)");

  /* Create all labels. */
  txt = .window.CreateControl (GStatic, x, 12, w, h,
      "Widgets that can display live data");
  format_label (txt, 18, 9, "Arial Bold", 0xee0000);
  txt = .window.CreateControl (GStatic, x, 43, w, h,
      "GStatic widget used for text display");
  format_label (txt, 14, 5, "Arial Bold", 0xee0000);
  txt = .window.CreateControl (GStatic, x, 83, w, h,
      "GProgressBarCtrl widget");
  format_label (txt, 14, 5, "Arial Bold", 0xee0000);
  txt = .window.CreateControl (GStatic, x, 133, w, h,
      "A custom progress bar made from 2 GStatic widgets");
  format_label (txt, 14, 5, "Arial Bold", 0xee0000);
txt = .window.CreateControl (GStatic, x, 183, w, h, "GListView widget");
format_label (txt, 14, 5, "Arial Bold", 0xee0000);

/* Create a text display for live Sine data. */
stSine = .window.CreateControl (GStatic, x, 60, w, h, "Sine");
.window.onChange (#$DataSim:Sine, `update (@stSine, this));

/* Create a progress bar widget to display live Sine data. */
pgSine = .window.CreateControl (GProgressBarCtrl, x, 100, w, h, "Sine");
pgSine.SetRange(0, 100);
.window.onChange (#$DataSim:Sine,
`((@pgSine).SetPos(($DataSim:Sine +.5) * 100)));
pgSine.SetPos(($DataSim:Sine +.5) * 100);

/* Create a custom progress bar out of two GStatic widgets
* for live Sine data. */
rectSine = .window.CreateControl (GStatic, x, 150, w, h, "", WS_BORDER);
rectSine.SetBackground(0, 0xdddddd, 0);
fillSine = .window.CreateControl (GStatic, x + 1, 151, w/2, h - 2, "");
fillSine.SetBackground(0, 0xbbbb00, 0);
.window.onChange (#$DataSim:Sine,
`((@fillSine).ResizeClient(xrect($DataSim:Sine, -.5, .5, @w), 18, 1)));

/* Create a list view widget for live Sine, Ramp,
* and Triangle data. */
lv = .window.CreateControl (GListViewCtrl, x, 200, w, h * 5, 
"lv", WS_BORDER);
lv.SetViewType (1);
lv.SetExtendedListViewStyle (LVS_EX_FULLROWSELECT |
LVS_EX_ONECLICKACTIVATE,
LVS_EX_FULLROWSELECT /*| 
LVS_EX_ONECLICKACTIVATE*/);
lv.InsertColumn (0, "Point", 0, 80, 0);
lv.InsertColumn (1, "Value", 0, 150, 0);
lv.InsertColumn (2, "Comments", 0, 100, 0);
lv.SetBackground (0, 0xffffff, 0);
.AddPoint ("Sine", "No change", "A sine wave.");
.AddPoint ("Ramp", "No change", "Ramping up.");
.AddPoint ("Triangle", "No change", "New for this list.");
Entering Data

This tutorial illustrates three ways to enter data from a window: in a text entry field, using radio buttons, or with a track bar. All of the widgets are linked to the `DataSim:Update-Frequency` point. They each change the value of the point. The text entry and track bar widgets also change to indicate the value of the point whenever the point changes.

Output

![Widgets for data entry](image)

Code

```javascript
require ("Application");
require ("WindowsSupport");

class MyDataEntry Application {
```

window.onChange (#$DataSim:Sine, `((@self).ChangeVal("Sine", $DataSim:Sine)));
.window.onChange (#$DataSim:Ramp, `((@self).ChangeVal("Ramp", $DataSim:Ramp)));
.window.onChange (#$DataSim:Triangle, `((@self).ChangeVal("Triangle", $DataSim:Triangle)));

.window.MessageHandler (WM_DESTROY, `destroy(@self));
.window.ShowWindow (SW_SHOW);
}

ApplicationSingleton (MyDataDisplay);```
window;
}

/* Sets a DataHub point to the position of a scrolling widget. */
method MyDataEntry.Scrolled (tb)
{
    $DataSim:UpdateFrequency = tb.GetPos();
}

/* Formats a label for convenience. */
method MyDataEntry.format_label (win, letterh, letterw, font, color)
{
    win.SetForeground (0, 0, color);
    win.SetFontEx (letterh, letterw,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, font);
}

/* Sets a DataHub point to the value of a widget. */
method MyDataEntry.SubmitEval (txt)
{
    local x;
    if (!undefined_p(x = eval_string (txt.GetWindowText())))
    {
        if (number_p(x) && (x >= 1) && (x <= 20))
            $DataSim:UpdateFrequency = x;
        else
            txt.SetWindowText("Try again");
    }
    else
        txt.SetWindowText("Try again");
    nil;
}

/* The 'main line' of the program. */
method MyDataEntry.constructor ()
{
    local rect = CreateRect (0, 0, 300, 300), txt, btn;
    local x = 20, y = 10, w = 300, h = 20;

    /* Create the window, set its size, position, and colors. */
    .window = new GWindow();
    .window.Create (0, rect, "Data Entry", WS_OVERLAPPEDWINDOW, 0);
    .window.MoveWindow(20, 200, 350, 285);
    .window.SetBackground(0, GetSysColor(COLOR_3DFACE), 0);
    .window.SetChildBackground(0, GetSysColor(COLOR_3DFACE), 0);
/* Allow for the possibility that DataSim isn't running or that UpdateFrequency isn't defined. */
$DataSim:UpdateFrequency := 5;
datahub_command("(cset DataSim:UpdateFrequency 5)" );

/* Create labels. */
txt = .window.CreateControl (GStatic, x, 12, w, h, 
   "Widgets for data entry");
.txt = .window.CreateControl (GStatic, x, 43, w, h, 
   "GEdit widget");
.txt = .window.CreateControl (GStatic, x, 85, w, h, 
   "Enter an update frequency between 1 and 20.");
.txt = .window.CreateControl (GStatic, x, 111, w, h, 
   "GGroupBox and GRadioButton widgets");
.txt = .window.CreateControl (GStatic, x, 188, w, h, 
   "GTrackBarCtrl widget");

/* Create a data entry field for Update Frequency. */
txt = .window.CreateControl (GEdit, x, 63, 60, 20, 
   string($DataSim:UpdateFrequency), 
   WS_BORDER);
.txt.SetBackground (0, GetSysColor (COLOR_WINDOW), 0);
.window onChange (#$DataSim:UpdateFrequency, 
   `(txt).SetWindowText (string(value)));

.btn = .window.CreateControl (GButton, x + 60 + 10, 
   63, 60, 20, "Submit");
.window.CommandHandler (BN_CLICKED, btn.GetDlgCtrlID(),
   `(self).SubmitEval (@txt));

/* Create a control group and two radio buttons for Update Frequency. */
b0 = .window.CreateControl (GGroupBox, x, 130, w, 40, 
   "Change Update Frequency");
b = b0.CreateControl (GRadioButton, 10, 15, 140, 25, 
   "UpdateFrequency = 1");
b2 = b0.CreateControl (GRadioButton, 150, 15, 140, 25, 
   "UpdateFrequency = 10");
b0.CommandHandler (BN_CLICKED, b.GetDlgCtrl1ID(),
   `(b0).IsDlgButtonChecked (@b).GetDlgCtrl1ID());
   `((@b0).IsDlgButtonChecked (@b).GetDlgCtrl1ID());
A complete demo program: WindowsExample.g

This tutorial demonstrates more widgets and useful features.
Output

![Output Image]

Code

```plaintext
require ("Application");
require ("WindowsSupport");

class WindowsExample Application
{
  menu_actions;
  menu_items;
  win;
}

method WindowsExample.constructor ()
{
  local win = new GWindow();
  local rect = CreateRect(10, 10, 400, 400);
  local b, tb, txt;

  win.Create (0, rect, "Monitor", WS_OVERLAPPEDWINDOW, 0);
  .win = win;

  win.SetBackground (0, GetSysColor (COLOR_3DFACE), 0);
  win.SetChildBackground (0, GetSysColor (COLOR_3DFACE), 0);
  win.MessageHandler (WM_DESTROY, `destroy (@self));
```
txt = win.CreateControl (GStatic, 20, 5, 350, 20,
        "Start DataSim to see live data here");
txt.SetForeground (0, 0, 0xff0000);
txt.SetFontEx (20, 8, 0, 0, 1, 0, 0, 0,
        0, 0, 0, 0, 0, "Times");
win.onChange (#$DataSim:Sine,
        `(@(txt).SetWindowText(format
        ("%a = %.4f",
        this, value))));

b0 = win.CreateControl (GGroupBox, 20, 30, 350, 40,
        "Radio Buttons");
win.AddControlResizeFlags (b0.GetDlgCtrlID(), DLSZ_SIZE_X);
b = b0.CreateControl (GRadioButton, 10, 15, 150, 25,
        "UpdateFrequency = 1");
b2 = b0.CreateControl (GRadioButton, 180, 15, 150, 25,
        "UpdateFrequency = 10");
b0.AddControlResizeFlags (b2.GetDlgCtrlID(), DLSZ_MOVE_X);

b0.CommandHandler (BN_CLICKED, b.GetDlgCtrlID(),
        `((@b0).IsDlgButtonChecked((@b).GetDlgCtrlID())
        == 0 ? nil :
        $DataSim:UpdateFrequency = 1));
b0.CommandHandler (BN_CLICKED, b2.GetDlgCtrlID(),
        `((@b0).IsDlgButtonChecked((@b2).GetDlgCtrlID())
        == 0 ? nil :
        $DataSim:UpdateFrequency = 10));

win.onChange (#$DataSim:UpdateFrequency,
        `(@(b).SetCheck (value == 1 ? 1 : 0));
win.onChange (#$DataSim:UpdateFrequency,
        `(@(b2).SetCheck (value == 10 ? 1 : 0));

txt = win.CreateControl (GStatic, 20, 80, 70, 25,
        "Transparency");
tb = .win.CreateControl (GTrackBarCtrl, 90, 80, 280, 25,
        "Transparency");
tb.SetRange (50, 255, 5);
tb.ModifyStyle (0, TBS_AUTOTICKS, 0);
tb.SetTicFreq (5);
tb.SetPos (255);
.win.MessageHandler (WM_HSCROLL,

// Make a field for entering Gamma commands
txt = win.CreateControl (GammaEntry, 20, 115, 350, 20,
"command");
txt.SetBackground (0, GetSysColor (COLOR_WINDOW), 0);
win.AddControlResizeFlags (txt.GetDlgItemID(), DLSZ_SIZE_X);
txt.SetWindowText ("Type a Gamma command, then press Enter");

// Make a tab control.

tab = win.CreateControl (GTabCtrl, 20, 150, 350, 200, "tabs");
tab.ModifyStyle (TCS_TABS |
                 TCS_RAGGEDRIGHT,
                 TCS_FOCUSONBUTTONDOWN |
                 TCS_FIXEDWIDTH);
tab.SetChildBackground (0, 0xc0ffff, 0);

tab.AddWindow (0, "Waveforms", .
               CreateDialog1 (tab.GetHwnd(), ";");
tab.AddWindow (1, "Parameters", .
               CreateDialog2 (tab.GetHwnd(), ";");
tab.AddWindow (2, "List", .
               CreateDialog3 (tab.GetHwnd(), ";");
win.AddControlResizeFlags (tab.GetDlgItemID(),
                          DLSZ_SIZE_X | DLSZ_SIZE_Y);
win.NotifyHandler (TCN_SELCHANGE,
                  tab.GetDlgItemID(),
                  @(tab).SelChange());
tab.SetCurSel (0);
.AddMenus();

win.ShowWindow (SW_SHOW);
}

method WindowsExample.destructor ()
{
  local  id, count, traymenu, i, item;

  try
  {
    with x in .menu_actions do
    remove_menu_action (car(x), cdr(x));
  }
  catch
  {
    prin ("Error: ", _last_error_, ";
      print_stack();
  }
}
with x in .menu_items do
{
try
{
id = car(x);
traymenu = get_tray_menu();
count = GetMenuItemCount (traymenu);
item = new MENUITEMINFO;
item.cbSize = 48;
item.fMask = MIIM_ID | MIIM_SUBMENU;

if (cdr(x)) // id is a submenu hmenu
{
for (i=0; i<count; i++)
{
if (GetMenuItemInfo
    (traymenu, i, 1, item)
    != 0
    && item.hSubMenu == id)
    DeleteMenu (traymenu,
               i,
               MF_BYPOSITION);
}
}
else // id is a menu item command id
{
for (i=0; i<count; i++)
{
if (GetMenuItemInfo
    (traymenu, i, 1, item)
    != 0
    && item.wID == id)
    DeleteMenu (traymenu,
               i,
               MF_BYPOSITION);
}
}
} catch
{
    prinl("Error: ", _last_error_, 
\n");
    print_stack();
}
if (!destroyed_p (.win))
    .win.DestroyWindow();
method WindowsExample.SetTransparency (alpha)
{
    // This bit of code makes the window translucent.
    .win.ModifyStyleEx (0, WS_EX_LAYERED);
    SetLayeredWindowAttributes (.win.GetHwnd(), 0, alpha, LWA_ALPHA);
}

/* ======= An entry field that evaluates its input as Gamma ======= */
class GammaEntry GEdit
{
}

method GammaEntry.constructor ()
{
    .MessageHandler (WM_CHAR, `(self).SubmitEval ());
}

method GammaEntry.Typing ()
{
    local str = .GetWindowText();
    princ (str, "\n");
}

method GammaEntry.SubmitEval ()
{
    if (wParam == VK_RETURN)
    {
        local str = .GetWindowText();
        princ (str, "\n");
        local x = eval_string (string(str,";"), t);
        princ ("    ");
        pretty_print (x);
        terpri();
    }
    nil;
}

/* ======= Print something when the user selects a file ======= */
/* This is an example of how to use the global hook on
   a file dialog to act on any file selection. */
method GFileDialog.OnSelChange ()
local buf = make_buffer(256);
    .GetFilePath (buf, 256);
    princ ("Selection changed: ", buffer_to_string(buf), 
*/
/* =========== Add sample menu items to the tray menu ========= */

MenuItemID := 10000;

method WindowsExample.SelectTrayMenuItem (id)
{
    local  win = new GColorDialog();
    princ ("HWND: ", win.m_hWnd, " = ", win.GetHwnd(), "\n");
    local  result = win.DoModal(0);
    if (result == 1)
    {
        princ ("You chose the color: ",
                hex(win.GetColor()), "\n");
    }
}

method WindowsExample.AddSubMenu (parent, pos, label)
{
    local submenu = CreatePopupMenu();
    InsertMenu (traymenu, pos, MF_BYPOSITION | MF_POPUP, 
                submenu, label);
    .menu_items = cons (cons (submenu, t), .menu_items);
    submenu;
}

method WindowsExample.AddMenuItem (parent, pos, label, code)
{
    local info = new MENUITEMINFO();

    info.cbSize = 48;
    info.fMask = MIIM_STRING | MIIM_FTYPE | MIIM_ID;
    info.fType = MFT_STRING;
    info.wID = ++MenuItemID;
    info.dwTypeData = label;
    InsertMenuItem (parent, pos, 1, info);
    local action = add_menu_action (MenuItemID, code);
    .menu_actions = cons (action, .menu_actions);
}

method WindowsExample.AddMenus ()
local traymenu = get_tray_menu();

if (traymenu != 0)
{
    local submenu = .AddSubMenu (traymenu, 5, "Monitor Functions");

    .AddMenuItem (submenu, -1, "Select Color", `(self).SelectTrayMenuItem (@MenuItemID+1));
    .AddMenuItem (submenu, -1, "Select File", `new GFileDialog (1).DoModal(0));
    .AddMenuItem (submenu, -1, "Select Folder", `new GFolderDialog ().DoModal(0));
}
else
{
}

/* ==== A dialog within the first tab in the tab control ==== */

class Dialog1 GWindow
{
    win;
stHeader;
stSine;
strRamp;
strTriangle;
strSquare;
}

method WindowsExample.CreateDialog1 (parent, str)
{
    local win = new Dialog1();
    win.Init (parent, str);
    win;
}

method Dialog1.Init (parent, str)
{
    local rect = CreateRect(10, 10, 10, 10);
    local row = 5, drow = 30, w = 290, h = 20;
.win = self;
.win.Create (parent, rect, "tab", WS_CHILDWINDOW, 0);

.stSine = .win.CreateControl (GStatic, 5, row, w, h, "Sine");
.win.onChange (#$DataSim:Sine, `WindowsExample_update (@.stSine, this));
WindowsExample_update (.stSine, #$DataSim:Sine);

.stRamp = .win.CreateControl (GStatic, 5, row += drow, w, h, "Ramp");
.win.onChange (#$DataSim:Ramp, `WindowsExample_update (@.stRamp, this));
WindowsExample_update (.stRamp, #$DataSim:Ramp);

.stTriangle = .win.CreateControl (GStatic, 5, row += drow, w, h, "Triangle");
.win.onChange (#$DataSim:Triangle, `WindowsExample_update (@.stTriangle, this));
WindowsExample_update (.stTriangle, #$DataSim:Triangle);

.stSquare = .win.CreateControl (GStatic, 5, row += drow, w, h, "Square");
.win.onChange (#$DataSim:Square, `WindowsExample_update (@.stSquare, this));
WindowsExample_update (.stSquare, #$DataSim:Square);

link = .win.CreateControl (GHyperLink, 5, row += drow, w, h, "http://www.cogent.ca");
link.SetHyperLink ("http://www.cogent.ca");
link.SetLabel ("Visit Cogent's web page");
link.SetLinkFont (GetStockObject (DEFAULT_GUI_FONT));
link.ShowWindow (SW_SHOW);

.win.onChange (#$DataSim:Square,
  `progn {
    if (value < 0)
    {
      (@win).SetBackground (0, 0xc0ffff, 0);
      (@win).SetChildBackground (0, 0xc0ffff, 0);
    }
    else
    {
      (@win).SetBackground (0, 0xffffff, 0);
      (@win).SetChildBackground (0, 0xffffff, 0);
    }
  }
(@win).Invalidate();
class Dialog2 GWindow
{
  win;
  stHeader;
  stAmplitude;
  stFrequency;
  stOffset;
  stUpdateFrequency;
}

method WindowsExample.CreateDialog2 (parent, str)
{
  local win = new Dialog2();
  win.Init (parent, str);
  win;
}

method Dialog2.Init (parent, str)
{
  local rect = CreateRect(10, 10, 10, 10);
  local row = 5, drow = 30, w = 290, h = 20;

  .win = self;
  .win.Create (parent, rect, "tab", WS_CHILDWINDOW, 0);

  .stAmplitude = .win.CreateControl (GStatic, 5, row,
    w, h, "Amplitude");
  .win.onChange (#$DataSim:Amplitude,
    'WindowsExample_update(@.stAmplitude, this));
  WindowsExample_update (.stAmplitude, #$DataSim:Amplitude);

  .stFrequency = .win.CreateControl (GStatic, 5, row += drow,
    w, h, "Frequency");
  .win.onChange (#$DataSim:Frequency,
    'WindowsExample_update(@.stFrequency, this));
  WindowsExample_update (.stFrequency, #$DataSim:Frequency);

  .stOffset = .win.CreateControl (GStatic, 5, row += drow,
    w, h, "Offset");
.win.onChange (#$DataSim:Offset,
  `WindowsExample_update(@.stOffset, this));
WindowsExample_update (.stOffset, #$DataSim:Offset);

.stUpdateFrequency = .win.CreateControl (GStatic, 5,
  row += drow,
  w, h,
  "UpdateFrequency");
.win.onChange (#$DataSim:UpdateFrequency,
  `WindowsExample_update(@.stUpdateFrequency,
    this));
WindowsExample_update (.stUpdateFrequency,
  #$DataSim:UpdateFrequency);

local tb = .win.CreateControl (GTrackBarCtrl, 5,
  row += drow, w, h,
  "FreqBar");
tb.SetRange (0, 20, 1);
tb.ModifyStyle (0, TBS_AUTOTICKS, 0);
tb.SetTicFreq (2);
.win.onChange (#$DataSim:UpdateFrequency,
  `(@tb).SetPos (value));
if (!undefined_p($DataSim:UpdateFrequency))
tb.SetPos ($DataSim:UpdateFrequency);
win;
}

method Dialog2.Scrolled (tb)
{
  $DataSim:UpdateFrequency = tb.GetPos();
}

/* ==== A dialog containing a List box ==== */

class Dialog3 GWindow
{
  win;
  lv;
}

method WindowsExample.CreateDialog3 (parent, str)
{
  local win = new Dialog3();
  win.Init (parent, str);
method Dialog3.AddPet (name, owner, species)
{
    local  n = .lv.GetItemCount();
    .lv.InsertItem (n, name);
    .lv.SetItemText (n, 1, owner);
    .lv.SetItemText (n, 2, species);
}

method Dialog3.Init (parent, str)
{
    local  rect = CreateRect(10, 10, 10, 10);
    local  w = 10, h = 10;

    .win = self;
    .win.SetBackground (0, GetSysColor (COLOR_3DFACE), 0);
    .win.Create (parent, rect, "tab", WS_CHILDWINDOW, 0);

    .lv = .CreateControl (GListViewCtrl, 0, 0, w, h, "lv");  
    .lv.SetViewType (1);
    .lv.SetExtendedListViewStyle (LVS_EX_FULLROWSELECT | LVS_EX_ONECLICKACTIVATE, LVS_EX_FULLROWSELECT | LVS_EX_ONECLICKACTIVATE);
    .lv.InsertColumn (0, "Pet", 0, 100, 0);
    .lv.InsertColumn (1, "Owner", 0, 100, 0);
    .lv.InsertColumn (2, "Species", 0, 100, 0);
    .lv.SetBackground (0, 0xffffff, 0);
    .AddPet ("Fluffy", "Harold", "cat");
    .AddPet ("Claws", "Gwen", "cat");
    .AddPet ("Buffy", "Harold", "dog");
    .AddPet ("Fang", "Benny", "dog");
    .AddPet ("Bowser", "Diane", "dog");
    .AddPet ("Chirpy", "Gwen", "bird");
    .AddPet ("Whistler", "Gwen", "bird");
    .AddPet ("Slim", "Benny", "snake");
    .AddPet ("Blob", "Benny", "slug");

    /* One way to modify the list after it has been created is to search for an entry by its column-zero text and then..."
modify the sub-items. */
local finfo = new LVFINDINFO();
finfo.flags = LVFI_STRING;
finfo.psz = "Chirpy";
local item = .lv.FindItem (finfo, 0);
.lv.SetItemText (item, 1, "Nobody");

.win.AddControlResizeFlags (.lv.GetDlgItemID(),
    DLSZ_SIZE_X | DLSZ_SIZE_Y);

.win.NotifyHandler (LVN_ITEMACTIVATE,
    .lv.GetDlgItemID(),
    `(self).SelectedItems(@(.lv)));

method Dialog3.SelectedItems (lv)
{
    local  i, n = lv.GetItemCount(), state, items;
    for (i=0; i<n; i++)
    {
        if ((state = lv.GetItemState (i, LVIS_SELECTED)) != 0)
            items = cons (i, items);
    }
    princ (reverse(items), "\n");
    reverse (items);
}

/* ================ Support functions ============== */

function WindowsExample_update (win, point)
{
    local  val, i;
    if (!undefined_p (val = eval(point)))
    {
        point = string (point);
        if ((i = strchr (point, ':')) != -1)
            point = substr (point, i+1, -1);
        win.SetWindowText (format ("%s = %.4f", point, val));
    }
}

/* ================ Run It =============== */
ApplicationSingleton (WindowsExample);
**Sample Code: ShowImage.g**

This tutorial demonstrates how to add a BMP image using the GStatic control.

```gcode
require ("Application");
require ("WindowsSupport");

class ShowImage Application
{
    window;
    imageFile = "c:/tmp/Capture.bmp";
    imageHandles;
}

/*
 * This creates a GStatic and populates it with an image from a file.
 * The image must be in Windows BMP format. You can specify a width
 * and height to force the image to be scaled to fit that size. If
 * you specify 0 for both width and height then the image will be
 * rendered in its natural size. Notice that we need to store the
 * image handle and then free it during the destructor.
 */
method ShowImage.createImage (parent, x, y, width, height, imageFile)
{
    local imageObj = parent.CreateControl(GStatic, x, y, width,
                                        height, ",", SS_BITMAP);
    local handle = LoadImage(0, imageFile, IMAGE_BITMAP,
                              width, height, LR_LOADFROMFILE);
    imageHandles = cons(handle, .imageHandles);
    imageObj.SendMessage(STM_SETIMAGE, IMAGE_BITMAP, handle);
    imageObj;
}

/* Write the 'main line' of the program here. */
method ShowImage.constructor ()
{
    local rect = CreateRect(10, 10, 400, 400);
    local win;

    win = new GWindow();
    win.Create(0, rect, "Test Window", WS_OVERLAPPEDWINDOW, 0);
    win.MessageHandler (WM_DESTROY,
```
```ruby
'(!destroy_p(@self) ?
  destroy(@self) : nil));

.createImage(win, 10, 10, 0, 0, .imageFile);

.window = win;
win.ShowWindow(SW_SHOW);
}

/* Any code to be run when the program gets shut down. We need to
delete the image handles. */
method ShowImage.destructor ()
{
  with handle in .imageHandles do
    DeleteObject(handle);
    if (instance_p(.window))
      destroy(.window);
  }

/* Start the program by instantiating the class. */
ApplicationSingleton (ShowImage);
```

**Sample Code: ListBoxExample.g**

This tutorial demonstrates the list box.

**Output**

![List Box Example](image)

**Code**

```ruby
/* All user scripts should derive from the base "Application" class */
```
```plaintext
require ("Application");
require ("WindowsSupport");

class ListBoxExample Application
{
  window;
}

// Fill the combo box with choices
method ListBoxExample.FillCombo(lb)
{
  local current = lb.GetWindowText();

  // Clear the combo box if we need to.
  lb.Clear();
  lb.ResetContent();

  // Add some options to the combo box
  lb.AddString("String 1");
  lb.AddString("String 2");

  // If there was an existing choice, reselect it
  lb.SelectString(-1, current);
}

// This is called when the combo box is selected
method ListBoxExample.ComboSelected(lb)
{
  princ ("You selected ", lb.GetWindowText(), 

}

/* Write the 'main line' of the program here. */
method ListBoxExample.constructor ()
{
  local win = new GWindow();
  local rect = CreateRect(10, 10, 400, 400);

  win.Create (0, rect, "ListBox Example", WS_OVERLAPPEDWINDOW, 0);
  win.MessageHandler (WM_DESTROY, `(!destroyed_p(@self) ?
     destroy(@self) : nil));
  .window = win;
  .window.SetBackground (0, GetSysColor (COLOR_3DFACE), 0);

  // ------- List box example starts here
  // Create a combo box. Use CBS_* to set the options.
```
// An editable combo uses CBS_DROPDOWN.
local lb = win.CreateControl (GComboBox, 5, 5, 200, 20, "ExampleListBox", CBS_DROPDOWNLIST | CBS_SORT);

// If the window has a different background, make sure the combo box stays white.
lb.SetChildBackground (0, GetSysColor (COLOR_WINDOW), 0);

// If we want to fill the combo when the person drops it down, do this
win.CommandHandler (CBN_DROPDOWN, lb, `(@self).FillCombo(@lb));
// Otherwise we could just do this to fill it once at the start
// .FillCombo(lb);

// Trigger an event when the user selects an option
win.CommandHandler (CBN_SELCHANGE, lb, `(@self).ComboSelected(@lb));

// ------- List box example ends here

win.ShowWindow (SW_SHOW);

/* Any code to be run when the program gets shut down. */
method ListBoxExample.destructor ()
{
    if (instance_p (.window))
        destroy (.window);
}

/* Start the program by instantiating the class. */
ApplicationSingleton (ListBoxExample);

Sample Code: TreeViewDemo.g

This tutorial demonstrates how to create a tree to view data.
Output

![TreeView Example](image)

Code

```javascript
/* All user scripts should derive from the base "Application" class */
require ("Application");
require ("WindowsSupport");

class TreeViewDemo Application
{
    window;
    tree;
    data = [
        ["top_level_1",
            ["level_1_1",
                ["level_2_1",
                    "level_3_1",
                    ["level_3_2", "level_4_1"],
                    ["level_3_3"
                ],
                ["level_2_2",
                    ["level_3_4", "level_3_5", "level_3_6"
                ]
            ],
            ["level_1_2",
                ["level_2_3", "level_2_4"
            ]
        ]
    ];

    /* Use methods to create functions outside the 'main line'. */
    method TreeViewDemo.FillTree (data)
    {
        local   root = .tree.GetRootItem();
        with descendants in data do
```
// Adding an item uses the InsertItem method.
method TreeViewDemo.FillTreeRecursive (parent, data)
{
    local   nodename = array_p(data) ? data[0] : data;
    local   htreeitem = .tree.InsertItem(nodename, 0, 0,
                                      parent, TVGN_LASTVISIBLE);
    local   i;

    if (array_p(data))
    {
        for (i=1; i<length(data); i++)
        {
            .FillTreeRecursive(htreeitem, data[i]);
        }
    }
}

// Find the full path name of the selected item in the tree
method TreeViewDemo.GetTreeSelection ()
{
    local       hitem = .tree.GetSelectedItem();
    .GetTreeItemFullname (hitem);
}

// Get just the text label of the given item
method TreeViewDemo.GetTreeItemText (hitem)
{
    local       item = new TVITEM();
    local       buf;

    item.hItem = hitem;
    item.mask = TVIF_TEXT;
    item.cchTextMax = 128;
    item.pszText = make_buffer(128);
    .tree.GetItem (item);
    buf = item.pszText;
    buffer_to_string (buf);
}

// Find the full path name of the given item. The path
// separator can be changed by modifying the / below
method TreeViewDemo.GetTreeItemFullname (hitem)
{
    local itemtext, pname, hparent, hgrandparent, sep = "/";
    if (hitem != 0)
    {
        itemtext = .GetTreeItemText (hitem);
        hparent = .tree.GetParentItem(hitem);
        if (hparent != 0)
        {
            hgrandparent = .tree.GetParentItem(hparent);
            // If you want to make the first separator different
            // from the rest, change it here.
            if (hgrandparent == 0)
                sep = "/";
        }

        pname = .GetTreeItemFullname (hparent);
        if (pname)
            string (pname, sep, itemtext);
        else
            itemtext;
    }
    else
        nil;
}

// Callback when an item is selected
method TreeViewDemo.cbItemSelected (itemname)
{
    princ ("You selected item: ", itemname, "\n");
}

// Callback before a branch in the tree is expanded or collapsed
method TreeViewDemo.cbPointExpanding ()
{
    local item = map_volatile_pointer (pnmh, NMTREEVIEW);
    local hitem = item.itemNew.hItem;
    local pointname = .GetTreeItemFullname (hitem);

    if (item.action == TVE_COLLAPSE)
    {
        princ ("Item: ", pointname, " is about to collapse\n");
    }
    else
    {
        princ ("Item: ", pointname, " is about to expand\n");
    }
/** Write the 'main line' of the program here. */
method TreeViewDemo.constructor ()
{
    local   win = new GWindow();
    local   rect = CreateRect(100, 100, 220, 220);
    local   clientrect = CreateRect(0,0,0,0);

    win.Create (0, rect, "TreeView Example", WS_OVERLAPPEDWINDOW, 0);
    win.MessageHandler (WM_DESTROY, `(!destroyed_p(@self) ?
        destroy(@self) : nil));
    .window = win;
    .window.SetBackground (0, GetSysColor (COLOR_3DFACE), 0);

    // Find the interior size of the window so we can size
    // the TreeView
    win.GetClientRect(clientrect);

    // ------- TreeView example starts here

    // Create a TreeView. Use TVS_* to set the options.

    // Callback after a branch is expanded or collapsed
    method TreeViewDemo.cbPointExpanded ()
    {
        local   item = map_volatile_pointer (pnmh, NMTREEVIEW);
        local   hitem = item.itemNew.hItem;
        local   pointname = .GetTreeItemFullname (hitem);

        if (item.action == TVE_COLLAPSE)
        {
            princ ("Item: ", pointname, " has collapsed\n");
        }
        else
        {
            princ ("Item: ", pointname, " has expanded\n");
        }
    }
local tv = win.CreateControl (GTreeViewCtrl, 5, 5,
    clientrect.right - 10,
    clientrect.bottom - 10,
    "ExampleTreeView",
    TVS_CHECKBOXES | TVS_HASLINES |
    TVS_LINESATROOT | TVS_HASBUTTONS |
    TVS_TRACKSELECT | TVS_NOHSCROLL);

.tree = tv;

// If the window has a different background, make sure
// the combo box stays white.
tv.SetChildBackground (0, GetSysColor (COLOR_WINDOW), 0);

// Fill the tree once at the start
.FillTree(.data);

// Trigger an event when the user selects an item
win.NotifyHandler (TVN_SELCHANGED, tv,
    `(@self).cbItemSelected((@self).GetTreeSelection()));

win.NotifyHandler (TVN_ITEMEXPANDING, tv,
    `(@self).cbPointExpanding());

win.NotifyHandler (TVN_ITEMEXPANDED, tv,
    `(@self).cbPointExpanded());

// There is no event in Windows for changing a check-box state
// in a TreeView !? Checkbox events are absurdly difficult to
// deal with in C++, and impossible in Gamma.

// You can add events to handle other use cases as well...
//win.NotifyHandler (NM_DBLCLK, tv,
//    `(@self).cbDoubleClick((@self).GetTreeSelection()));
//tv.MessageHandler (WM_RBUTTONDOWN, `((@self).cbRightClick()));

// Reize the TreeView when the window resizes
win.AddControlResizeFlags (tv, DLSZ_SIZE_X | DLSZ_SIZE_Y);

// ------ TreeView example ends here

win.ShowWindow (SW_SHOW);
}

/* Any code to be run when the program gets shut down. */
method TreeViewDemo.destructor ()
{
    if (instance_p(.window))
destroy (\.window);
}

/* Start the program by instantiating the class. If your
 * constructor code does not create a persistent reference to
 * the instance (self), then it will be destroyed by the
 * garbage collector soon after creation. If you do not want
 * this to happen, assign the instance to a global variable, or
 * create a static data member in your class to which you assign
 * 'self' during the construction process. ApplicationSingleton()
 * does this for you automatically. */
ApplicationSingleton (TreeViewDemo);

Sample Code: Browse DataHub Points using TreeViewExample.g

This tutorial demonstrates how to create a tree to browse points in the DataHub.

Output

![Treeview Example]

Code

/* All user scripts should derive from the base "Application" class */

require ("Application");
require ("WindowsSupport");

/*
 * The real implementation of the point browser is in
 * c:\program files\cogent\opc datahub\require\GPointBrowser.g
 */

require ("GPointBrowser");
class TreeViewExample Application
{
    window;
    gtree;
}

/*
 * Specialize the GPointBrowser object so that we get our own
 * callbacks when certain events occur in the point tree. The
 * default event handling will still be processed to fill the
 * tree as the user traverses it.
 */

class MyGPointBrowser GPointBrowser
{
    ...
}

method MyGPointBrowser.OnSelect (pointname)
{
    princ ("Selected: ", pointname, "\n");
}

method MyGPointBrowser.OnDoubleClick(pointname)
{
    princ ("Double Click: ", pointname, "\n");
}

method MyGPointBrowser.OnRightClick(pointname)
{
    princ ("Right Click: ", pointname, "\n");
}

method MyGPointBrowser.OnExpanded (pointname)
{
    princ ("Expanded: ", pointname, "\n");
}

method MyGPointBrowser.OnCollapsed (pointname)
{
    princ ("Collapsed: ", pointname, "\n");
}

/* Write the 'main line' of the program here. */
method TreeViewExample.constructor ()
{
    local rect = CreateRect (0, 0, 300, 300);
window = new GWindow();

window.Create (0, rect, "GPointBrowser Test", WS_OVERLAPPEDWINDOW, 0);
window.CenterWindow();
window.GetClientRect (rect);
gtree = window.CreateControl (MyGPointBrowser, 0, 0, rect.right - rect.left, rect.bottom - rect.top, "Hello world", SS_CENTER);
gtree.CenterWindow();
gtree.debugging = t;
window.MessageHandler (WM_DESTROY, `instance_p(@self) ? destroy(@self) : nil);
window.AddControlResizeFlags (.gtree, DLSZ_SIZE_X | DLSZ_SIZE_Y);
window.ShowWindow (SW_SHOW);

/* Any code to be run when the program gets shut down. */
method TreeViewExample.destructor ()
{
    if (instance_p(.window))
        destroy (.window);
}

/* Start the program by instantiating the class. */
ApplicationSingleton (TreeViewExample);
Widgets
G3StateButton

G3StateButton — a 3-state button.

Synopsis

```cpp
class G3StateButton GButton
{
    m_hWnd;
}
```

Base Classes

```cpp
GWindowBase --> GButton <-- G3StateButton
```

Description

This widget is a superclass that provides the BS_3STATE button style of a CButton.

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

```cpp
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to C3StateButton::Create.
```

This class also inherits the functions of GButton and GWindowBase.
**GAnimateCtrl**

GAnimateCtrl — an animation control.

**Synopsis**

```cpp
class GAnimateCtrl : GWindowBase
{
  m_hWnd;
}
```

**Base Classes**

GWindowBase <-- GAnimateCtrl

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- Close ()
  Corresponds to CAnimateCtrl::Close.
- Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  Corresponds to CAnimateCtrl::Create.
- Play (nFrom, nTo, nRep)
  Corresponds to CAnimateCtrl::Play.
- Seek (nTo)
  Corresponds to CAnimateCtrl::Seek.
- Stop ()
  Corresponds to CAnimateCtrl::Stop.

This class also inherits the functions of GWindowBase.
**GBitmapButton**

GBitmapButton — a bitmap button.

**Synopsis**

```cpp
class GBitmapButton : public GWindowBase
{
    m_hWnd;
}
```

**Base Classes**

GWindowBase <-- GBitmapButton

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **AddControlResizeFlags (ctrlid, flags)**
  Corresponds to `CBitmapButton::AddControlResizeFlags`.

- **Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)**
  Corresponds to `CBitmapButton::Create`.

- **EndResizeGroup ()**
  Corresponds to `CBitmapButton::EndResizeGroup`.

- **StartResizeGroup ()**
  Corresponds to `CBitmapButton::StartResizeGroup`.

This class also inherits the functions of GWindowBase.
**Widgets**

**GButton**

GButton — a button.

**Synopsis**

```cpp
class GButton GWindowBase
{
    m_hWnd;
}
```

**Base Classes**

GWindowBase <- GButton

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- AddControlResizeFlags (ctrlid, flags)
  - Corresponds to CButton::AddControlResizeFlags.

- Click ()
  - Corresponds to CButton::Click.

- Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  - Corresponds to CButton::Create.

- EndResizeGroup ()
  - Corresponds to CButton::EndResizeGroup.

- GetButtonStyle ()
  - Corresponds to CButton::GetButtonStyle.

- GetCheck ()
  - Corresponds to CButton::GetCheck.

- GetIcon ()
  - Corresponds to CButton::GetIcon.

- GetIdealSize (lpSize)
  - Corresponds to CButton::GetIdealSize.

- GetState ()
  - Corresponds to CButton::GetState.

- GetTextMargin (lpRect)
  - Corresponds to CButton::GetTextMargin.
Widgets

SetButtonStyle (args?...)
   Corresponds to CButton::SetButtonStyle.

SetCheck (nCheck)
   Corresponds to CButton::SetCheck.

SetIcon (hIcon)
   Corresponds to CButton::SetIcon.

SetState (bHighlight)
   Corresponds to CButton::SetState.

SetTextMargin (lpRect)
   Corresponds to CButton::SetTextMargin.

StartResizeGroup ()
   Corresponds to CButton::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GCheckBox

GCheckBox — a check box.

Synopsis

class GCheckBox
GButton
{
    m_hWnd;
}

Base Classes

GWindowBase <-- GButton <-- GCheckBox

Description

This widget is a superclass that provides the BS_CHECKBOX button style of a CButton.

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CCheckBox::Create.

This class also inherits the functions of GButton and GWindowBase.
GCheckListViewCtrl

GCheckListViewCtrl — purpose is not yet documented.

Synopsis

class GCheckListViewCtrl GWindowBase
{
   m_hWnd;
}

Base Classes

GWindowBase <-- GCheckListViewCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
   Corresponds to CCheckListViewCtrl::AddControlResizeFlags.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
   Corresponds to CCheckListViewCtrl::Create.
EndResizeGroup ()
   Corresponds to CCheckListViewCtrl::EndResizeGroup.
StartResizeGroup ()
   Corresponds to CCheckListViewCtrl::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GColorDialog

GColorDialog — a color-selection dialog box.

Synopsis

class GColorDialog
{
    m_hWnd;
}

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

DoModal (args?...)

    Corresponds to CColorDialog::DoModal.

GetColor ()

    Corresponds to CColorDialog::GetColor.

GetHwnd ()

    Corresponds to CColorDialog::GetHwnd.

SetCurrentColor (clr)

    Corresponds to CColorDialog::SetCurrentColor.

This class also inherits the functions of
**GComboBox**

GComboBox — a list box combined with a static or edit control.

**Synopsis**

```cpp
class GComboBox GWindowBase
{
  m_hWnd;
}
```

**Base Classes**

GWindowBase <-- GComboBox

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- `AddControlResizeFlags (ctrlid, flags)`
  - Corresponds to CComboBox::AddControlResizeFlags.

- `AddString (lpszString)`
  - Corresponds to CComboBox::AddString.

- `Clear ()`
  - Corresponds to CComboBox::Clear.

- `Copy ()`
  - Corresponds to CComboBox::Copy.

- `Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)`
  - Corresponds to CComboBox::Create.

- `Cut ()`
  - Corresponds to CComboBox::Cut.

- `DeleteString (nIndex)`
  - Corresponds to CComboBox::DeleteString.

- `Dir (attr, lpszWildCard)`
  - Corresponds to CComboBox::Dir.

- `EndResizeGroup ()`
  - Corresponds to CComboBox::EndResizeGroup.

- `FindString (nStartAfter, lpszString)`
  - Corresponds to CComboBox::FindString.`
FindStringExact (nIndexStart, lpszFind)
Corresponds to CComboBox::FindStringExact.

GetComboBoxInfo (pComboBoxInfo)
Corresponds to CComboBox::GetComboBoxInfo.

GetCount ()
Corresponds to CComboBox::GetCount.

GetCurSel ()
Corresponds to CComboBox::GetCurSel.

GetDroppedControlRect (lprect)
Corresponds to CComboBox::GetDroppedControlRect.

GetDroppedState ()
Corresponds to CComboBox::GetDroppedState.

GetDroppedWidth ()
Corresponds to CComboBox::GetDroppedWidth.

GetEditSel ()
Corresponds to CComboBox::GetEditSel.

GetExtendedUI ()
Corresponds to CComboBox::GetExtendedUI.

GetHorizontalExtent ()
Corresponds to CComboBox::GetHorizontalExtent.

GetItemData (nIndex)
Corresponds to CComboBox::GetItemData.

GetItemHeight (nIndex)
Corresponds to CComboBox::GetItemHeight.

GetLBText (nIndex, lpszText)
Corresponds to CComboBox::GetLBText.

GetLBTextLen (nIndex)
Corresponds to CComboBox::GetLBTextLen.

GetLocale ()
Corresponds to CComboBox::GetLocale.

GetMinVisible ()
Corresponds to CComboBox::GetMinVisible.

GetTopIndex ()
Corresponds to CComboBox::GetTopIndex.

InitStorage (nItems, nBytes)
Corresponds to CComboBox::InitStorage.

InsertString (nIndex, lpszString)
Corresponds to CComboBox::InsertString.
LimitText (nMaxChars)
  Corresponds to CComboBox::LimitText.

Paste ()
  Corresponds to CComboBox::Paste.

ResetContent ()
  Corresponds to CComboBox::ResetContent.

SelectString (nStartAfter, lpszString)
  Corresponds to CComboBox::SelectString.

SetCurSel (nSelect)
  Corresponds to CComboBox::SetCurSel.

SetDroppedWidth (nWidth)
  Corresponds to CComboBox::SetDroppedWidth.

SetEditSel (nStartChar, nEndChar)
  Corresponds to CComboBox::SetEditSel.

SetExtendedUI (args?...)
  Corresponds to CComboBox::SetExtendedUI.

SetHorizontalExtent (nExtent)
  Corresponds to CComboBox::SetHorizontalExtent.

SetItemData (nIndex, dwItemData)
  Corresponds to CComboBox::SetItemData.

SetItemHeight (nIndex, cyItemHeight)
  Corresponds to CComboBox::SetItemHeight.

SetLocale (nNewLocale)
  Corresponds to CComboBox::SetLocale.

SetMinVisible (nMinVisible)
  Corresponds to CComboBox::SetMinVisible.

SetTopIndex (nIndex)
  Corresponds to CComboBox::SetTopIndex.

ShowDropDown (args?...)
  Corresponds to CComboBox::ShowDropDown.

StartResizeGroup ()
  Corresponds to CComboBox::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GComboBoxEx

GComboBoxEx — a combo box with support for image lists.

Synopsis

```cpp
class GComboBoxEx GComboBox
{
    m_hWnd;
}
```

Base Classes

```
GWindowBase <-- GComboBox <-- GComboBoxEx
```

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **Create** (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  - Corresponds to `CComboBoxEx::Create`.
- **DeleteItem** (nIndex)
  - Corresponds to `CComboBoxEx::DeleteItem`.
- **GetExtendedStyle** ()
  - Corresponds to `CComboBoxEx::GetExtendedStyle`.
- **GetItem** (pCBItem)
  - Corresponds to `CComboBoxEx::GetItem`.
- **GetUnicodeFormat** ()
  - Corresponds to `CComboBoxEx::GetUnicodeFormat`.
- **HasEditChanged** ()
  - Corresponds to `CComboBoxEx::HasEditChanged`.
- **InsertItem** (lpcCBItem)
  - Corresponds to `CComboBoxEx::InsertItem`.
- **SetExtendedStyle** (dwExMask, dwExStyle)
  - Corresponds to `CComboBoxEx::SetExtendedStyle`.
- **SetItem** (lpcCBItem)
  - Corresponds to `CComboBoxEx::SetItem`.
- **SetUnicodeFormat** (args?...)  
  - Corresponds to `CComboBoxEx::SetUnicodeFormat`.
This class also inherits the functions of \texttt{GComboBox} and \texttt{GWindowBase}. 
GDateTimePickerCtrl

GDateTimePickerCtrl — a way to enter a date and time, based on CDateTimeCtrl.

Synopsis

class GDateTimePickerCtrl GWindowBase
{
    m_hWnd;
}

Base Classes

GWindowBase <- GDateTimePickerCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  
  Corresponds to CDateTimePickerCtrl::Create.

- GetMonthCalColor (nColorType)
  
  Corresponds to CDateTimePickerCtrl::GetMonthCalColor.

- GetRange (lpSysTimeArray)
  
  Corresponds to CDateTimePickerCtrl::GetRange.

- GetSystemTime (lpSysTime)
  
  Corresponds to CDateTimePickerCtrl::GetSystemTime.

- SetFormat (lpszFormat)
  
  Corresponds to CDateTimePickerCtrl::SetFormat.

- SetMonthCalColor (nColorType, clr)
  
  Corresponds to CDateTimePickerCtrl::SetMonthCalColor.

- SetMonthCalFont (args?)
  
  Corresponds to CDateTimePickerCtrl::SetMonthCalFont.

- SetRange (dwFlags, lpSysTimeArray)
  
  Corresponds to CDateTimePickerCtrl::SetRange.

- SetSystemTime (dwFlags, lpSysTime)
  
  Corresponds to CDateTimePickerCtrl::SetSystemTime.

This class also inherits the functions of GWindowBase.
GDialog

GDialog — a dialog box.

Synopsis

class GDialog GWindowBase
{
    m_hWnd;
}

Base Classes

GWindowBase <- GDialog

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (args?...)
    Corresponds to CDialog::Create.
DoModal (args?...)
    Corresponds to CDialog::DoModal.
EndDialog (nRetCode)
    Corresponds to CDialog::EndDialog.
MapDialogRect (lpRect)
    Corresponds to CDialog::MapDialogRect.

This class also inherits the functions of GWindowBase.
**GDragListBox**

GDragListBox — allows listed items in a list box to be moved.

**Synopsis**

```cpp
class GDragListBox GListBox
{
    m_hWnd;
}
```

**Base Classes**

GWindowBase ← GListBox ← GDragListBox

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  Corresponds to CDragListBox::Create.

DrawInsert (nItem)
  Corresponds to CDragListBox::DrawInsert.

LBItemFromPt (args?)
  Corresponds to CDragListBox::LBItemFromPt.

MakeDragList ()
  Corresponds to CDragListBox::MakeDragList.

This class also inherits the functions of GListBox and GWindowBase.
**GEdit**

GEdit — a text-entry box.

**Synopsis**

class GEdit GWWindowBase
{
  m_hWnd;
}

**Base Classes**

GWWindowBase <-- GEdit

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- AddControlResizeFlags (ctrlid, flags)
  Corresponds to CEdit::AddControlResizeFlags.
- AppendText (args?...)
  Corresponds to CEdit::AppendText.
- CanUndo ()
  Corresponds to CEdit::CanUndo.
- CharFromPos (args?...)
  Corresponds to CEdit::CharFromPos.
- Clear ()
  Corresponds to CEdit::Clear.
- Copy ()
  Corresponds to CEdit::Copy.
- Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  Corresponds to CEdit::Create.
- Cut ()
  Corresponds to CEdit::Cut.
- EmptyUndoBuffer ()
  Corresponds to CEdit::EmptyUndoBuffer.
- EndResizeGroup ()
  Corresponds to CEdit::EndResizeGroup.
FmtLines (bAddEOL)
    Corresponds to CEdit::FmtLines.
GetFirstVisibleLine ()
    Corresponds to CEdit::GetFirstVisibleLine.
GetHandle ()
    Corresponds to CEdit::GetHandle.
GetImeStatus (uStatus)
    Corresponds to CEdit::GetImeStatus.
GetLimitText ()
    Corresponds to CEdit::GetLimitText.
GetLine (args?...)
    Corresponds to CEdit::GetLine.
GetLineCount ()
    Corresponds to CEdit::GetLineCount.
GetMargins ()
    Corresponds to CEdit::GetMargins.
GetModify ()
    Corresponds to CEdit::GetModify.
GetPasswordChar ()
    Corresponds to CEdit::GetPasswordChar.
GetRect (lpRect)
    Corresponds to CEdit::GetRect.
GetSel (args?...)
    Corresponds to CEdit::GetSel.
GetThumb ()
    Corresponds to CEdit::GetThumb.
HideBalloonTip ()
    Corresponds to CEdit::HideBalloonTip.
InsertText (args?...)
    Corresponds to CEdit::InsertText.
LimitText (args?...)
    Corresponds to CEdit::LimitText.
LineFromChar (args?...)
    Corresponds to CEdit::LineFromChar.
LineIndex (args?...)
    Corresponds to CEdit::LineIndex.
LineLength (args?...)
    Corresponds to CEdit::LineLength.
Widgets

LineScroll (args?...)
Corresponds to CEdit::LineScroll.

Paste ()
Corresponds to CEdit::Paste.

PosFromChar (nChar)
Corresponds to CEdit::PosFromChar.

ReplaceSel (args?...)
Corresponds to CEdit::ReplaceSel.

Scroll (nScrollAction)
Corresponds to CEdit::Scroll.

ScrollCaret ()
Corresponds to CEdit::ScrollCaret.

SetHandle (hBuffer)
Corresponds to CEdit::SetHandle.

SetImeStatus (uStatus, uData)
Corresponds to CEdit::SetImeStatus.

SetLimitText (nMax)
Corresponds to CEdit::SetLimitText.

SetMargins (nLeft, nRight)
Corresponds to CEdit::SetMargins.

SetModify (args?...)
Corresponds to CEdit::SetModify.

SetPasswordChar (ch)
Corresponds to CEdit::SetPasswordChar.

SetReadOnly (args?...)
Corresponds to CEdit::SetReadOnly.

SetRect (lpRect)
Corresponds to CEdit::SetRect.

SetRectNP (lpRect)
Corresponds to CEdit::SetRectNP.

SetSel (args?...)
Corresponds to CEdit::SetSel.

SetSelAll (args?...)
Corresponds to CEdit::SetSelAll.

SetSelNone (args?...)
Corresponds to CEdit::SetSelNone.

SetTabStops (args?...)
Corresponds to CEdit::SetTabStops.
ShowBalloonTip (pEditBaloonTip)
    Corresponds to CEdit::ShowBalloonTip.

StartResizeGroup ()
    Corresponds to CEdit::StartResizeGroup.

Undo ()
    Corresponds to CEdit::Undo.

This class also inherits the functions of GWindowBase.
**GFileDialog**

GFileDialog — a dialog box for opening and saving files.

**Synopsis**

```cpp
class GFileDialog
{
    m_bOpenFileDialog;
    m_ofn;
    m_szFileName;
    m_szFileTitle;
}
```

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- `DoModal (args??...)`
  Corresponds to `CFileDialog::DoModal`.
- `EndDialog (args??...)`
  Corresponds to `CFileDialog::EndDialog`.
- `GetFilePath (lpstrFilePath, nLength)`
  Corresponds to `CFileDialog::GetFilePath`.
- `GetFolderPath (lpstrFolderPath, nLength)`
  Corresponds to `CFileDialog::GetFolderPath`.
- `GetHwnd ()`
  Corresponds to `CFileDialog::GetHwnd`.
- `GetReadOnlyPref ()`
  Corresponds to `CFileDialog::GetReadOnlyPref`.
- `GetSpec (lpstrSpec, nLength)`
  Corresponds to `CFileDialog::GetSpec`.
- `HideControl (nCtrlID)`
  Corresponds to `CFileDialog::HideControl`.
- `SetControlText (nCtrlID, lpstrText)`
  Corresponds to `CFileDialog::SetControlText`.
- `SetDefExt (lpstrExt)`
  Corresponds to `CFileDialog::SetDefExt`.
This class also inherits the functions of
GFlatScrollBar

GFlatScrollBar — a scrollbar with an enhanced interface, based on FlatScrollBar.

Synopsis

```cpp
class GFlatScrollBar GWindowBase
{
    m_hWnd;
}
```

Base Classes

GWindowBase <-- GFlatScrollBar

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
   Corresponds to CFlatScrollBar::AddControlResizeFlags.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
   Corresponds to CFlatScrollBar::Create.
EndResizeGroup ()
   Corresponds to CFlatScrollBar::EndResizeGroup.
StartResizeGroup ()
   Corresponds to CFlatScrollBar::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GFOLDERDIALOG

GFOLDERDIALOG — a dialog box to manage folders, based on Dialog.

Synopsis

class GFolderDialog GDialog
{
   m_hWnd;
   m_szFolderDisplayName;
   m_szFolderPath;
}

Base Classes

GWindowBase <-> GDialog <-> GFOLDERDIALOG

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

DoModal (args?...)
   Corresponds to CFOLDERDIALOG::DoModal.

This class also inherits the functions of GDialog and GWindowBase.
GFontDialog

GFontDialog — a dialog box to manage fonts, based on Font Dialog.

Synopsis

```cpp
class GFontDialog GDialog
{
   m_hWnd;
}
```

Base Classes

GWindowBase <-- GDialog <-- GFontDialog

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

DoModal (args?...)

Corresponds to CFontDialog::DoModal.

This class also inherits the functions of GDialog and GWindowBase.
GGroupBox

GGroupBox — a rectangle that groups controls, based on Group Box.

Synopsis

```cpp
class GGroupBox GButton
{
    m_hWnd;
}
```

Base Classes

GWindowBase <-- GButton <-- GGroupBox

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
Corresponds to CGroupBox::Create.

This class also inherits the functions of GButton and GWindowBase.
GHeaderCtrl

GHeaderCtrl — a header for columns of text or numbers.

Synopsis

```cpp
class GHeaderCtrl : GWindowBase
{
    m_hWnd;
}
```

Base Classes

GWindowBase <- GHeaderCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- `AddControlResizeFlags (ctrlid, flags)`
  Corresponds to `CHeaderCtrl::AddControlResizeFlags`

- `ClearAllFilters ()`
  Corresponds to `CHeaderCtrl::ClearAllFilters`

- `ClearFilter (nColumn)`
  Corresponds to `CHeaderCtrl::ClearFilter`

- `Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)`
  Corresponds to `CHeaderCtrl::Create`

- `DeleteItem (nIndex)`
  Corresponds to `CHeaderCtrl::DeleteItem`

- `EditFilter (nColumn, bDiscardChanges)`
  Corresponds to `CHeaderCtrl::EditFilter`

- `EndResizeGroup ()`
  Corresponds to `CHeaderCtrl::EndResizeGroup`

- `GetBitmapMargin ()`
  Corresponds to `CHeaderCtrl::GetBitmapMargin`

- `GetItem (nIndex, pHeaderItem)`
  Corresponds to `CHeaderCtrl::GetItem`

- `GetItemCount ()`
  Corresponds to `CHeaderCtrl::GetItemCount`
GetItemRect (nIndex, lpItemRect)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::GetItemRect}.
GetUnicodeFormat ()
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::GetUnicodeFormat}.
HitTest (lpHitTestInfo)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::HitTest}.
InsertItem (nIndex, phdi)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::InsertItem}.
Layout (pHeaderLayout)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::Layout}.
OrderToIndex (nOrder)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::OrderToIndex}.
SetBitmapMargin (nWidth)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::SetBitmapMargin}.
SetFilterChangeTimeout (dwTimeOut)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::SetFilterChangeTimeout}.
SetHotDivider (bPos, dwInputValue)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::SetHotDivider}.
SetItem (nIndex, pHeaderItem)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::SetItem}.
SetUnicodeFormat (args?...)
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::SetUnicodeFormat}.
StartResizeGroup ()
\hspace{1em} \textbf{Corresponds to} \texttt{CHeaderCtrl::StartResizeGroup}.

This class also inherits the functions of \texttt{GWindowBase}.
**GHotKeyCtrl**

GHotKeyCtrl — allows the creation of a hot key.

**Synopsis**

```cpp
class GHotKeyCtrl : GWindowBase
{
    m_hWnd;
}
```

**Base Classes**

**GWindowBase** ← GHotKeyCtrl

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)**
  - Corresponds to CHotKeyCtrl::Create.

- **GetHotKey (args?...)**
  - Corresponds to CHotKeyCtrl::GetHotKey.

- **SetHotKey (wVirtualKeyCode, wModifiers)**
  - Corresponds to CHotKeyCtrl::SetHotKey.

- **SetRules (wInvalidComb, wModifiers)**
  - Corresponds to CHotKeyCtrl::SetRules.

This class also inherits the functions of GWindowBase.
**GHyperLink**

GHyperLink — displays a link to a web page, based on HyperLink.

**Synopsis**

```cpp
class GHyperLink : public GWindowBase {
    m_clrLink;
    m_clrVisited;
    m_dwExtendedStyle;
    m_hCursor;
    m_hFont;
    m_hFontNormal;
    m_hWnd;
    m_lpstrHyperLink;
    m_lpstrLabel;
    m_rcLink;
};
```

**Base Classes**

GWindowBase <-- GHyperLink

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- `Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)`
  Corresponds to CHyperLink::Create.
- `GetHyperLink (lpstrBuffer, nLength)`
  Corresponds to CHyperLink::GetHyperLink.
- `GetHyperLinkExtendedStyle ()`
  Corresponds to CHyperLink::GetHyperLinkExtendedStyle.
- `GetIdealHeight ()`
  Corresponds to CHyperLink::GetIdealHeight.
- `GetIdealSize (args?...)`
  Corresponds to CHyperLink::GetIdealSize.
- `GetLabel (lpstrBuffer, nLength)`
  Corresponds to CHyperLink::GetLabel.
GetLinkFont ()
  \textbf{Corresponds to} CHyperLink::GetLinkFont.
GetToolTipText (lpstrBuffer, nLength)
  \textbf{Corresponds to} CHyperLink::GetToolTipText.
IsCommandButton ()
  \textbf{Corresponds to} CHyperLink::IsCommandButton.
IsNotUnderlined ()
  \textbf{Corresponds to} CHyperLink::IsNotUnderlined.
IsNotifyButton ()
  \textbf{Corresponds to} CHyperLink::IsNotifyButton.
IsUnderlineHover ()
  \textbf{Corresponds to} CHyperLink::IsUnderlineHover.
IsUnderlined ()
  \textbf{Corresponds to} CHyperLink::IsUnderlined.
IsUsingTags ()
  \textbf{Corresponds to} CHyperLink::IsUsingTags.
IsUsingTagsBold ()
  \textbf{Corresponds to} CHyperLink::IsUsingTagsBold.
IsUsingToolTip ()
  \textbf{Corresponds to} CHyperLink::IsUsingToolTip.
Navigate ()
  \textbf{Corresponds to} CHyperLink::Navigate.
SetHyperLink (lpstrLink)
  \textbf{Corresponds to} CHyperLink::SetHyperLink.
SetHyperLinkExtendedStyle (args?...)
  \textbf{Corresponds to} CHyperLink::SetHyperLinkExtendedStyle.
SetLabel (lpstrLabel)
  \textbf{Corresponds to} CHyperLink::SetLabel.
SetLinkFont (hFont)
  \textbf{Corresponds to} CHyperLink::SetLinkFont.
SetToolTipText (lpstrToolTipText)
  \textbf{Corresponds to} CHyperLink::SetToolTipText.

This class also inherits the functions of \texttt{GWindowBase}.
GIPAddressCtrl

GIPAddressCtrl — an entry field for an IP address.

Synopsis

class GIPAddressCtrl GWindowBase
{
   m_hWnd;
}

Base Classes

GWindowBase <-- GIPAddressCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

ClearAddress ()
    Corresponds to CIPAddressCtrl::ClearAddress.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CIPAddressCtrl::Create.
IsBlank ()
    Corresponds to CIPAddressCtrl::IsBlank.
SetAddress (dwAddress)
    Corresponds to CIPAddressCtrl::SetAddress.
SetFocus (nField)
    Corresponds to CIPAddressCtrl::SetFocus.
SetRange (args?)
    Corresponds to CIPAddressCtrl::SetRange.

This class also inherits the functions of GWindowBase.
GLinkCtrl

GLinkCtrl — embeds a hypertext link in a window.

Synopsis

```cpp
class GLinkCtrl : GWindowBase
{
    m_hWnd;
}
```

Base Classes

GWindowBase <-- GLinkCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CLinkCtrl::Create.
GetIdealHeight ()
    Corresponds to CLinkCtrl::GetIdealHeight.
GetItem (pLItem)
    Corresponds to CLinkCtrl::GetItem.
HitTest (pLHitTestInfo)
    Corresponds to CLinkCtrl::HitTest.
SetItem (pLItem)
    Corresponds to CLinkCtrl::SetItem.

This class also inherits the functions of GWindowBase.
**GListBox**

GListBox — a box with a list of items.

**Synopsis**

```c
class GListBox GWindowBase
{
   m_hWnd;
}
```

**Base Classes**

GWindowBase <- GListBox

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- `AddControlResizeFlags (ctrlid, flags)`  
  Corresponds to CListBox::AddControlResizeFlags.

- `AddFile (lpstrFileName)`  
  Corresponds to CListBox::AddFile.

- `AddString (lpszItem)`  
  Corresponds to CListBox::AddString.

- `Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)`  
  Corresponds to CListBox::Create.

- `DeleteString (nIndex)`  
  Corresponds to CListBox::DeleteString.

- `Dir (attr, lpszWildCard)`  
  Corresponds to CListBox::Dir.

- `EndResizeGroup ()`  
  Corresponds to CListBox::EndResizeGroup.

- `FindString (nStartAfter, lpszItem)`  
  Corresponds to CListBox::FindString.

- `FindStringExact (nIndexStart, lpszFind)`  
  Corresponds to CListBox::FindStringExact.

- `GetAnchorIndex ()`  
  Corresponds to CListBox::GetAnchorIndex.
GetCaretIndex ()
   Corresponds to CListBox::GetCaretIndex.

GetCount ()
   Corresponds to CListBox::GetCount.

GetCurSel ()
   Corresponds to CListBox::GetCurSel.

GetHorizontalExtent ()
   Corresponds to CListBox::GetHorizontalExtent.

GetItemData (nIndex)
   Corresponds to CListBox::GetItemData.

GetItemHeight (nIndex)
   Corresponds to CListBox::GetItemHeight.

GetItemRect (nIndex, lpRect)
   Corresponds to CListBox::GetItemRect.

GetListBoxInfo ()
   Corresponds to CListBox::GetListBoxInfo.

GetLocale ()
   Corresponds to CListBox::GetLocale.

GetSel (nIndex)
   Corresponds to CListBox::GetSel.

GetSelCount ()
   Corresponds to CListBox::GetSelCount.

GetText (nIndex, lpszBuffer)
   Corresponds to CListBox::GetText.

GetTextLen (nIndex)
   Corresponds to CListBox::GetTextLen.

GetTopIndex ()
   Corresponds to CListBox::GetTopIndex.

InitStorage (nItems, nBytes)
   Corresponds to CListBox::InitStorage.

InsertString (nIndex, lpszItem)
   Corresponds to CListBox::InsertString.

ResetContent ()
   Corresponds to CListBox::ResetContent.

SelItemRange (bSelect, nFirstItem, nLastItem)
   Corresponds to CListBox::SelItemRange.

SelectString (nStartAfter, lpszItem)
   Corresponds to CListBox::SelectString.
SetAnchorIndex (nIndex)
Corresponds to CListBox::SetAnchorIndex.

SetCaretIndex (args?)
Corresponds to CListBox::SetCaretIndex.

SetColumnWidth (cxWidth)
Corresponds to CListBox::SetColumnWidth.

SetCount (cItems)
Corresponds to CListBox::SetCount.

SetCurSel (nSelect)
Corresponds to CListBox::SetCurSel.

SetHorizontalExtent (cxExtent)
Corresponds to CListBox::SetHorizontalExtent.

SetItemData (nIndex, dwItemData)
Corresponds to CListBox::SetItemData.

SetItemHeight (nIndex, cyItemHeight)
Corresponds to CListBox::SetItemHeight.

SetLocale (nNewLocale)
Corresponds to CListBox::SetLocale.

SetSel (args?)
Corresponds to CListBox::SetSel.

SetTabStops (args?)
Corresponds to CListBox::SetTabStops.

SetTopIndex (nIndex)
Corresponds to CListBox::SetTopIndex.

StartResizeGroup ()
Corresponds to CListBox::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GListViewCtrl

GListViewCtrl — determines how lists are displayed, based on ListView Control.

Synopsis

class GListViewCtrl GWindowBase
{
  m_hWnd;
}

Base Classes

GWindowBase <- GListViewCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

addColumn (args?...)  
Corresponds to CListViewCtrl::AddColumn.

AddControlResizeFlags (ctrlid, flags)  
Corresponds to CListViewCtrl::AddControlResizeFlags.

addItem (args?...)  
Corresponds to CListViewCtrl::AddItem.

ApproximateViewRect (args?...)  
Corresponds to CListViewCtrl::ApproximateViewRect.

Arrange (nCode)  
Corresponds to CListViewCtrl::Arrange.

CancelEditLabel ()  
Corresponds to CListViewCtrl::CancelEditLabel.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)  
Corresponds to CListViewCtrl::Create.

DeleteAllItems ()  
Corresponds to CListViewCtrl::DeleteAllItems.

DeleteColumn (nCol)  
Corresponds to CListViewCtrl::DeleteColumn.

DeleteItem (nItem)  
Corresponds to CListViewCtrl::DeleteItem.
EnableGroupView (bEnable)
    Corresponds to CListViewCtrl::EnableGroupView.

EndResizeGroup ()
    Corresponds to CListViewCtrl::EndResizeGroup.

EnsureVisible (nItem, bPartialOK)
    Corresponds to CListViewCtrl::EnsureVisible.

FindItem (pFindInfo, nStart)
    Corresponds to CListViewCtrl::FindItem.

GetBkColor ()
    Corresponds to CListViewCtrl::GetBkColor.

GetBkImage (plvbk1)
    Corresponds to CListViewCtrl::GetBkImage.

GetCallbackMask ()
    Corresponds to CListViewCtrl::GetCallbackMask.

GetCheckState (nIndex)
    Corresponds to CListViewCtrl::GetCheckState.

GetColumn (nCol, pColumn)
    Corresponds to CListViewCtrl::GetColumn.

GetColumnWidth (nCol)
    Corresponds to CListViewCtrl::GetColumnWidth.

GetCountPerPage ()
    Corresponds to CListViewCtrl::GetCountPerPage.

GetExtendedListViewStyle ()
    Corresponds to CListViewCtrl::GetExtendedListViewStyle.

GetGroupInfo (nGroupID, pGroup)
    Corresponds to CListViewCtrl::GetGroupInfo.

GetGroupMetrics (pGroupMetrics)
    Corresponds to CListViewCtrl::GetGroupMetrics.

GetHotCursor ()
    Corresponds to CListViewCtrl::GetHotCursor.

GetHotItem ()
    Corresponds to CListViewCtrl::GetHotItem.

GetHoverTime ()
    Corresponds to CListViewCtrl::GetHoverTime.

GetISearchString (lpstr)
    Corresponds to CListViewCtrl::GetISearchString.

GetInsertMarkColor ()
    Corresponds to CListViewCtrl::GetInsertMarkColor.
GetInsertMarkRect (lpRect)
    Corresponds to CListViewCtrl::GetInsertMarkRect.
GetItem (pItem)
    Corresponds to CListViewCtrl::GetItem.
GetItemCount ()
    Corresponds to CListViewCtrl::GetItemCount.
GetItemData (nItem)
    Corresponds to CListViewCtrl::GetItemData.
GetItemPosition (nItem, lpPoint)
    Corresponds to CListViewCtrl::GetItemPosition.
GetItemRect (nItem, lpRect, nCode)
    Corresponds to CListViewCtrl::GetItemRect.
GetItemSpacing (args?...)
    Corresponds to CListViewCtrl::GetItemSpacing.
GetItemState (nItem, nMask)
    Corresponds to CListViewCtrl::GetItemState.
GetItemText (args?...)
    Corresponds to CListViewCtrl::GetItemText.
GetNextItem (nItem, nFlags)
    Corresponds to CListViewCtrl::GetNextItem.
GetOrigin (lpPoint)
    Corresponds to CListViewCtrl::GetOrigin.
GetOutlineColor ()
    Corresponds to CListViewCtrl::GetOutlineColor.
GetSelectedColumn ()
    Corresponds to CListViewCtrl::GetSelectedColumn.
GetSelectedCount ()
    Corresponds to CListViewCtrl::GetSelectedCount.
GetSelectedIndex ()
    Corresponds to CListViewCtrl::GetSelectedIndex.
GetSelectedItem (pItem)
    Corresponds to CListViewCtrl::GetSelectedItem.
GetSelectionMark ()
    Corresponds to CListViewCtrl::GetSelectionMark.
GetStringWidth (lpsz)
    Corresponds to CListViewCtrl::GetStringWidth.
GetSubItemRect (nItem, nSubItem, nFlag, lpRect)
    Corresponds to CListViewCtrl::GetSubItemRect.
GetTextBkColor ()
    Corresponds to CListViewCtrl::GetTextBkColor.
GetTextColor ()
    Corresponds to CListViewCtrl::GetTextColor.
GetTileInfo (pTileInfo)
    Corresponds to CListViewCtrl::GetTileInfo.
GetTileViewInfo (pTileViewInfo)
    Corresponds to CListViewCtrl::GetTileViewInfo.
GetTopIndex ()
    Corresponds to CListViewCtrl::GetTopIndex.
GetUnicodeFormat ()
    Corresponds to CListViewCtrl::GetUnicodeFormat.
GetView ()
    Corresponds to CListViewCtrl::GetView.
GetViewRect (lpRect)
    Corresponds to CListViewCtrl::GetViewRect.
GetViewType ()
    Corresponds to CListViewCtrl::GetViewType.
GetWorkAreas (nWorkAreas, lpRect)
    Corresponds to CListViewCtrl::GetWorkAreas.
HasGroup (nGroupID)
    Corresponds to CListViewCtrl::HasGroup.
HitTest (args?...)
    Corresponds to CListViewCtrl::HitTest.
InsertColumn (args?...)
    Corresponds to CListViewCtrl::InsertColumn.
InsertGroup (nItem, pGroup)
    Corresponds to CListViewCtrl::InsertGroup.
InsertGroupSorted (pInsertGroupSorted)
    Corresponds to CListViewCtrl::InsertGroupSorted.
InsertItem (args?...)
    Corresponds to CListViewCtrl::InsertItem.
IsGroupViewEnabled ()
    Corresponds to CListViewCtrl::IsGroupViewEnabled.
MapIDToIndex (uID)
    Corresponds to CListViewCtrl::MapIDToIndex.
MapIndexToID (nIndex)
    Corresponds to CListViewCtrl::MapIndexToID.
MoveGroup (nGroupID, nItem)
   Corresponds to CListViewCtrl::MoveGroup.
MoveItemToGroup (nItem, nGroupID)
   Corresponds to CListViewCtrl::MoveItemToGroup.
RedrawItems (nFirst, nLast)
   Corresponds to CListViewCtrl::RedrawItems.
RemoveAllGroups ()
   Corresponds to CListViewCtrl::RemoveAllGroups.
RemoveGroup (nGroupID)
   Corresponds to CListViewCtrl::RemoveGroup.
Scroll (size)
   Corresponds to CListViewCtrl::Scroll.
SelectItem (nIndex)
   Corresponds to CListViewCtrl::SelectItem.
SetBkColor (cr)
   Corresponds to CListViewCtrl::SetBkColor.
SetBkImage (plvbki)
   Corresponds to CListViewCtrl::SetBkImage.
SetCallbackMask (nMask)
   Corresponds to CListViewCtrl::SetCallbackMask.
SetCheckState (nItem, bCheck)
   Corresponds to CListViewCtrl::SetCheckState.
SetColumn (nCol, pColumn)
   Corresponds to CListViewCtrl::SetColumn.
SetColumnWidth (nCol, cx)
   Corresponds to CListViewCtrl::SetColumnWidth.
SetExtendedListViewStyle (args?...)
   Corresponds to CListViewCtrl::SetExtendedListViewStyle.
SetGroupInfo (nGroupID, pGroup)
   Corresponds to CListViewCtrl::SetGroupInfo.
SetGroupMetrics (pGroupMetrics)
   Corresponds to CListViewCtrl::SetGroupMetrics.
SetHotCursor (hHotCursor)
   Corresponds to CListViewCtrl::SetHotCursor.
SetHotItem (nIndex)
   Corresponds to CListViewCtrl::SetHotItem.
SetHoverTime (dwHoverTime)
   Corresponds to CListViewCtrl::SetHoverTime.
SetIconSpacing (cx, cy)
    Corresponds to CListViewCtrl::SetIconSpacing.
SetInfoTip (pSetInfoTip)
    Corresponds to CListViewCtrl::SetInfoTip.
SetInsertMarkColor (clr)
    Corresponds to CListViewCtrl::SetInsertMarkColor.
SetItem (args?...)
    Corresponds to CListViewCtrl::SetItem.
SetItemCount (nItems)
    Corresponds to CListViewCtrl::SetItemCount.
SetItemCountEx (nItems, dwFlags)
    Corresponds to CListViewCtrl::SetItemCountEx.
SetItemData (nItem, dwData)
    Corresponds to CListViewCtrl::SetItemData.
SetItemPosition (args?...)
    Corresponds to CListViewCtrl::SetItemPosition.
SetItemState (args?...)
    Corresponds to CListViewCtrl::SetItemState.
SetItemText (nItem, nSubItem, lpszText)
    Corresponds to CListViewCtrl::SetItemText.
SetOutlineColor (clr)
    Corresponds to CListViewCtrl::SetOutlineColor.
SetSelectedColumn (nColumn)
    Corresponds to CListViewCtrl::SetSelectedColumn.
SetSelectionMark (nIndex)
    Corresponds to CListViewCtrl::SetSelectionMark.
SetTextBkColor (cr)
    Corresponds to CListViewCtrl::SetTextBkColor.
SetTextColor (cr)
    Corresponds to CListViewCtrl::SetTextColor.
SetTileInfo (pTileInfo)
    Corresponds to CListViewCtrl::SetTileInfo.
SetTileViewInfo (pTileViewInfo)
    Corresponds to CListViewCtrl::SetTileViewInfo.
SetUnicodeFormat (args?...)
    Corresponds to CListViewCtrl::SetUnicodeFormat.
SetView (dwView)
    Corresponds to CListViewCtrl::SetView.
SetViewType (dwType)
    Corresponds to CListViewCtrl::SetViewType.

SetWorkAreas (nWorkAreas, lpRect)
    Corresponds to CListViewCtrl::SetWorkAreas.

SortGroups (args?...)
    Corresponds to CListViewCtrl::SortGroups.

StartResizeGroup ()
    Corresponds to CListViewCtrl::StartResizeGroup.

SubItemHitTest (lpInfo)
    Corresponds to CListViewCtrl::SubItemHitTest.

Update (nItem)
    Corresponds to CListViewCtrl::Update.

This class also inherits the functions of GWindowBase.
GMonthCalendarCtrl

GMonthCalendarCtrl — a control for setting dates and times, based on MonthCalendar Control.

Synopsis

```cpp
class GMonthCalendarCtrl : GWindowBase
{
    m_hWnd;
}
```

Base Classes

GWindowBase <- GMonthCalendarCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CMonthCalendarCtrl::Create.
GetColor (nColorType)
    Corresponds to CMonthCalendarCtrl::GetColor.
GetCurSel (lpSysTime)
    Corresponds to CMonthCalendarCtrl::GetCurSel.
GetFirstDayOfWeek (args?...)
    Corresponds to CMonthCalendarCtrl::GetFirstDayOfWeek.
GetMaxSelCount ()
    Corresponds to CMonthCalendarCtrl::GetMaxSelCount.
GetMinReqRect (lpRectInfo)
    Corresponds to CMonthCalendarCtrl::GetMinReqRect.
GetMonthDelta ()
    Corresponds to CMonthCalendarCtrl::GetMonthDelta.
GetMonthRange (args?...)
    Corresponds to CMonthCalendarCtrl::GetMonthRange.
GetRange (lprgSysTimeArray)
    Corresponds to CMonthCalendarCtrl::GetRange.
GetSelRange (lprgSysTimeArray)
    Corresponds to CMonthCalendarCtrl::GetSelRange.
GetToday (lpSysTime)
    Corresponds to CMonthCalendarCtrl::GetToday.
GetUnicodeFormat ()
    Corresponds to CMonthCalendarCtrl::GetUnicodeFormat.
SetColor (nColorType, clr)
    Corresponds to CMonthCalendarCtrl::SetColor.
SetCurSel (lpSysTime)
    Corresponds to CMonthCalendarCtrl::SetCurSel.
SetFirstDayOfWeek (args?...)
    Corresponds to CMonthCalendarCtrl::SetFirstDayOfWeek.
SetMaxSelCount (nMax)
    Corresponds to CMonthCalendarCtrl::SetMaxSelCount.
SetMonthDelta (nDelta)
    Corresponds to CMonthCalendarCtrl::SetMonthDelta.
SetRange (dwFlags, lprgSysTimeArray)
    Corresponds to CMonthCalendarCtrl::SetRange.
SetSelRange (lprgSysTimeArray)
    Corresponds to CMonthCalendarCtrl::SetSelRange.
SetToday (lpSysTime)
    Corresponds to CMonthCalendarCtrl::SetToday.
SetUnicodeFormat (args?...)
    Corresponds to CMonthCalendarCtrl::SetUnicodeFormat.

This class also inherits the functions of GWindowBase.
**GPageSetupDialog**

GPageSetupDialog — a control for print page setup options.

**Synopsis**

```cpp
class GPageSetupDialog : public GDialog
{
    m_hWnd;
}
```

**Base Classes**

GWindowBase <- GDialog <- GPageSetupDialog

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

`DoModal (args?...)`

Corresponds to `CPageSetupDialog::DoModal`.

This class also inherits the functions of `GDialog` and `GWindowBase`. 
GPagerCtrl

GPagerCtrl — a container for a window without enough area to display its content.

Synopsis

```cpp
class GPagerCtrl GWindowBase
{
    m_hWnd;
}
```

Base Classes

```cpp
GWindowBase <-- GPagerCtrl
```

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- `AddControlResizeFlags (ctrlid, flags)`
  Corresponds to `CPagerCtrl::AddControlResizeFlags`.

- `Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)`
  Corresponds to `CPagerCtrl::Create`.

- `EndResizeGroup ()`
  Corresponds to `CPagerCtrl::EndResizeGroup`.

- `ForwardMouse (args?)...`
  Corresponds to `CPagerCtrl::ForwardMouse`.

- `GetBkColor ()`
  Corresponds to `CPagerCtrl::GetBkColor`.

- `GetBorder ()`
  Corresponds to `CPagerCtrl::GetBorder`.

- `GetButtonSize ()`
  Corresponds to `CPagerCtrl::GetButtonSize`.

- `GetButtonState (nButton)`
  Corresponds to `CPagerCtrl::GetButtonState`.

- `GetPos ()`
  Corresponds to `CPagerCtrl::GetPos`.

- `RecalcSize ()`
  Corresponds to `CPagerCtrl::RecalcSize`.

SetBkColor (clrBk)  
Corresponds to CPagerCtrl::SetBkColor.

SetBorder (nBorderSize)  
Corresponds to CPagerCtrl::SetBorder.

SetButtonSize (nButtonSize)  
Corresponds to CPagerCtrl::SetButtonSize.

SetChild (hWndChild)  
Corresponds to CPagerCtrl::SetChild.

SetPos (nPos)  
Corresponds to CPagerCtrl::SetPos.

StartResizeGroup ()  
Corresponds to CPagerCtrl::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GPrintDialog

GPrintDialog — a print dialog window.

Synopsis

class GPrintDialog GDial{G{}
   m_hWnd;
}

Base Classes

GWindowBase <-> GDial <-> GPrintDialog

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

DoModal (args?...)
   Corresponds to CPrintDialog::DoModal.

This class also inherits the functions of GDial and GWindowBase.
GProgressBarCtrl

GProgressBarCtrl — a rectangle that fills over time, based on ProgressBar Control.

Synopsis

class GProgressBarCtrl GWWindowBase
{
    m_hWnd;
}

Base Classes

GWWindowBase <-> GProgressBarCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
    Corresponds to CProgressBarCtrl::AddControlResizeFlags.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CProgressBarCtrl::Create.
EndResizeGroup ()
    Corresponds to CProgressBarCtrl::EndResizeGroup.
GetPos ()
    Corresponds to CProgressBarCtrl::GetPos.
GetRangeLimit (bLimit)
    Corresponds to CProgressBarCtrl::GetRangeLimit.
OffsetPos (nPos)
    Corresponds to CProgressBarCtrl::OffsetPos.
SetBarColor (clr)
    Corresponds to CProgressBarCtrl::SetBarColor.
SetBkColor (clr)
    Corresponds to CProgressBarCtrl::SetBkColor.
SetMarquee (args?...)
    Corresponds to CProgressBarCtrl::SetMarquee.
SetPos (nPos)
    Corresponds to CProgressBarCtrl::SetPos.
SetRange (nLower, nUpper)
  Corresponds to CProgressBarCtrl::SetRange.

SetRange32 (nMin, nMax)
  Corresponds to CProgressBarCtrl::SetRange32.

SetStep (nStep)
  Corresponds to CProgressBarCtrl::SetStep.

StartResizeGroup ()
  Corresponds to CProgressBarCtrl::StartResizeGroup.

StepIt ()
  Corresponds to CProgressBarCtrl::StepIt.

This class also inherits the functions of GWindowBase.
GRadioButton

GRadioButton — a radio button, based on CButton.

Synopsis

class GRadioButton 
{
    m_hWnd;
}

Base Classes

GWindowBase <- GButton <- GRadioButton

Description

This widget is a superclass that provides the BS_RADIOBUTTON button style of a CButton.

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
   Corresponds to CRadioButton::Create.

This class also inherits the functions of GButton and GWindowBase.
**GReBarCtrl**

GReBarCtrl — a container for a child window.

**Synopsis**

```cpp
class GReBarCtrl GWindowBase
{
    m_hWnd;
}
```

**Base Classes**

GWindowBase <- GReBarCtrl

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **AddControlResizeFlags** (ctrlid, flags)
  
  Corresponds to CReBarCtrl::AddControlResizeFlags.
- **BeginDrag** (args?...)
  
  Corresponds to CReBarCtrl::BeginDrag.
- **Create** (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  
  Corresponds to CReBarCtrl::Create.
- **DeleteBand** (nBand)
  
  Corresponds to CReBarCtrl::DeleteBand.
- **DragMove** (args?...)
  
  Corresponds to CReBarCtrl::DragMove.
- **EndDrag**()
  
  Corresponds to CReBarCtrl::EndDrag.
- **EndResizeGroup**()
  
  Corresponds to CReBarCtrl::EndResizeGroup.
- **GetBandBorders** (nBand, lpRect)
  
  Corresponds to CReBarCtrl::GetBandBorders.
- **GetBandCount**()
  
  Corresponds to CReBarCtrl::GetBandCount.
- **GetBandInfo** (nBand, lprbbi)
  
  Corresponds to CReBarCtrl::GetBandInfo.
GetBarHeight ()
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetBarHeight}.

GetBarInfo (lprbi)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetBarInfo}.

GetBkColor ()
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetBkColor}.

GetColorScheme (lpColorScheme)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetColorScheme}.

GetPalette ()
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetPalette}.

GetRect (nBand, lpRect)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetRect}.

GetRowCount ()
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetRowCount}.

GetRowHeight (nBand)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetRowHeight}.

GetTextColor ()
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetTextColor}.

GetUnicodeFormat ()
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::GetUnicodeFormat}.

HitTest (lprbht)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::HitTest}.

IdToIndex (uBandID)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::IdToIndex}.

InsertBand (nBand, lprbbi)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::InsertBand}.

LockBands (bLock)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::LockBands}.

MaximizeBand (nBand)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::MaximizeBand}.

MinimizeBand (nBand)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::MinimizeBand}.

MoveBand (nBand, nNewPos)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::MoveBand}.

PushChevron (nBand, lAppValue)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::PushChevron}.

SetBandInfo (nBand, lprbbi)
\hspace{1em} \textbf{Corresponds to} \texttt{CReBarCtrl::SetBandInfo}.
Widgets

SetBarInfo (lprbi)
    Corresponds to CReBarCtrl::SetBarInfo.
SetBkColor (clr)
    Corresponds to CReBarCtrl::SetBkColor.
SetColorScheme (lpColorScheme)
    Corresponds to CReBarCtrl::SetColorScheme.
SetPalette (hPalette)
    Corresponds to CReBarCtrl::SetPalette.
SetTextColor (clr)
    Corresponds to CReBarCtrl::SetTextColor.
SetToolTips (hwndToolTip)
    Corresponds to CReBarCtrl::SetToolTips.
SetUnicodeFormat (args?...)
    Corresponds to CReBarCtrl::SetUnicodeFormat.
ShowBand (nBand, bShow)
    Corresponds to CReBarCtrl::ShowBand.
SizeToRect (lpRect)
    Corresponds to CReBarCtrl::SizeToRect.
StartResizeGroup ()
    Corresponds to CReBarCtrl::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GRichEditCtrl

GRichEditCtrl — a text entry and editing window.

Synopsis

class GRichEditCtrl GWindowBase
{
    m_hWnd;
}

Base Classes

GWindowBase <~ GRichEditCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
    Corresponds to CRichEditCtrl::AddControlResizeFlags.

AppendText (args?...)
    Corresponds to CRichEditCtrl::AppendText.

CanPaste (args?...)
    Corresponds to CRichEditCtrl::CanPaste.

CanRedo ()
    Corresponds to CRichEditCtrl::CanRedo.

CanUndo ()
    Corresponds to CRichEditCtrl::CanUndo.

CharFromPos (pt)
    Corresponds to CRichEditCtrl::CharFromPos.

Clear ()
    Corresponds to CRichEditCtrl::Clear.

Copy ()
    Corresponds to CRichEditCtrl::Copy.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CRichEditCtrl::Create.

Cut ()
    Corresponds to CRichEditCtrl::Cut.
DisplayBand (pDisplayRect)

Corresponds to CRichEditCtrl::DisplayBand.

EmptyUndoBuffer ()

Corresponds to CRichEditCtrl::EmptyUndoBuffer.

EndResizeGroup ()

Corresponds to CRichEditCtrl::EndResizeGroup.

FindTextA (args?)

Corresponds to CRichEditCtrl::FindTextA.

FindWordBreak (nCode, nStartChar)

Corresponds to CRichEditCtrl::FindWordBreak.

FormatRange (args?)

Corresponds to CRichEditCtrl::FormatRange.

GetAutoURLDetect ()

Corresponds to CRichEditCtrl::GetAutoURLDetect.

GetDefaultCharFormat (args?)

Corresponds to CRichEditCtrl::GetDefaultCharFormat.

GetEventMask ()

Corresponds to CRichEditCtrl::GetEventMask.

GetFirstVisibleLine ()

Corresponds to CRichEditCtrl::GetFirstVisibleLine.

GetLimitText ()

Corresponds to CRichEditCtrl::GetLimitText.

GetLine (args?)

Corresponds to CRichEditCtrl::GetLine.

GetLineCount ()

Corresponds to CRichEditCtrl::GetLineCount.

GetModify ()

Corresponds to CRichEditCtrl::GetModify.

GetOptions ()

Corresponds to CRichEditCtrl::GetOptions.

GetParaFormat (args?)

Corresponds to CRichEditCtrl::GetParaFormat.

GetRect (lpRect)

Corresponds to CRichEditCtrl::GetRect.

GetRedoName ()

Corresponds to CRichEditCtrl::GetRedoName.

GetSel (args?)

Corresponds to CRichEditCtrl::GetSel.
GetSelText (lpBuf)
    Corresponds to CRichEditCtrl::GetSelText.
GetSelectionCharFormat (args?...)
    Corresponds to CRichEditCtrl::GetSelectionCharFormat.
GetSelectionType ()
    Corresponds to CRichEditCtrl::GetSelectionType.
GetTextEx (args?...)
    Corresponds to CRichEditCtrl::GetTextEx.
GetTextLength ()
    Corresponds to CRichEditCtrl::GetTextLength.
GetTextLengthEx (args?...)
    Corresponds to CRichEditCtrl::GetTextLengthEx.
GetTextMode ()
    Corresponds to CRichEditCtrl::GetTextMode.
GetTextRange (args?...)
    Corresponds to CRichEditCtrl::GetTextRange.
GetUndoName ()
    Corresponds to CRichEditCtrl::GetUndoName.
HideSelection (args?...)
    Corresponds to CRichEditCtrl::HideSelection.
InsertText (args?...)
    Corresponds to CRichEditCtrl::InsertText.
LimitText (args?...)
    Corresponds to CRichEditCtrl::LimitText.
LineFromChar (nIndex)
    Corresponds to CRichEditCtrl::LineFromChar.
LineIndex (args?...)
    Corresponds to CRichEditCtrl::LineIndex.
LineLength (args?...)
    Corresponds to CRichEditCtrl::LineLength.
LineScroll (args?...)
    Corresponds to CRichEditCtrl::LineScroll.
Paste ()
    Corresponds to CRichEditCtrl::Paste.
PasteSpecial (args?...)
    Corresponds to CRichEditCtrl::PasteSpecial.
PosFromChar (nChar)
    Corresponds to CRichEditCtrl::PosFromChar.
Redo ()
  Corresponds to CRichEditCtrl::Redo.

ReplaceSel (args?...)
  Corresponds to CRichEditCtrl::ReplaceSel.

RequestResize ()
  Corresponds to CRichEditCtrl::RequestResize.

ScrollCaret ()
  Corresponds to CRichEditCtrl::ScrollCaret.

SetAutoURLDetect (args?...)
  Corresponds to CRichEditCtrl::SetAutoURLDetect.

SetBackgroundColor (args?...)
  Corresponds to CRichEditCtrl::SetBackgroundColor.

SetCharFormat (args?...)
  Corresponds to CRichEditCtrl::SetCharFormat.

SetDefaultCharFormat (args?...)
  Corresponds to CRichEditCtrl::SetDefaultCharFormat.

SetEventMask (dwEventMask)
  Corresponds to CRichEditCtrl::SetEventMask.

SetModify (args?...)
  Corresponds to CRichEditCtrl::SetModify.

SetOptions (wOperation, dwOptions)
  Corresponds to CRichEditCtrl::SetOptions.

SetPalette (hPalette)
  Corresponds to CRichEditCtrl::SetPalette.

SetParaFormat (args?...)
  Corresponds to CRichEditCtrl::SetParaFormat.

SetReadOnly (args?...)
  Corresponds to CRichEditCtrl::SetReadOnly.

SetRect (lpRect)
  Corresponds to CRichEditCtrl::SetRect.

SetSel (args?...)
  Corresponds to CRichEditCtrl::SetSel.

SetSelAll ()
  Corresponds to CRichEditCtrl::SetSelAll.

SetSelNone ()
  Corresponds to CRichEditCtrl::SetSelNone.

SetSelectionCharFormat (args?...)
  Corresponds to CRichEditCtrl::SetSelectionCharFormat.
SetTargetDevice (hDC, cxLineWidth)
  Corresponds to CRichEditCtrl::SetTargetDevice.
SetTextMode (enumTextMode)
  Corresponds to CRichEditCtrl::SetTextMode.
SetUndoLimit (uUndoLimit)
  Corresponds to CRichEditCtrl::SetUndoLimit.
SetWordCharFormat (args?...)
  Corresponds to CRichEditCtrl::SetWordCharFormat.
ShowScrollBar (args?...)
  Corresponds to CRichEditCtrl::ShowScrollBar.
StartResizeGroup ()
  Corresponds to CRichEditCtrl::StartResizeGroup.
StopGroupTyping ()
  Corresponds to CRichEditCtrl::StopGroupTyping.
StreamIn (uFormat, es)
  Corresponds to CRichEditCtrl::StreamIn.
StreamOut (uFormat, es)
  Corresponds to CRichEditCtrl::StreamOut.
Undo ()
  Corresponds to CRichEditCtrl::Undo.

This class also inherits the functions of GWindowBase.
GScrollBar

GScrollBar — a scroll-bar control.

Synopsis

```cpp
class GScrollBar GWindowBase
{
  
  m_hWnd;
}
```

Base Classes

```cpp
GWindowBase <-- GScrollBar
```

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  Corresponds to CScrollBar::Create.

EnableScrollBar (args?...)
  Corresponds to CScrollBar::EnableScrollBar.

GetScrollBarInfo (pScrollBarInfo)
  Corresponds to CScrollBar::GetScrollBarInfo.

GetScrollInfo (lpScrollInfo)
  Corresponds to CScrollBar::GetScrollInfo.

GetScrollLimit ()
  Corresponds to CScrollBar::GetScrollLimit.

GetScrollPos ()
  Corresponds to CScrollBar::GetScrollPos.

SetScrollInfo (args?...)
  Corresponds to CScrollBar::SetScrollInfo.

SetScrollPos (args?...)
  Corresponds to CScrollBar::SetScrollPos.

SetScrollRange (args?...)
  Corresponds to CScrollBar::SetScrollRange.

ShowScrollBar (args?...)
  Corresponds to CScrollBar::ShowScrollBar.
This class also inherits the functions of `GWindowBase`.
GStatic

GStatic — a static control for text strings, rectangles, bitmaps, etc.

Synopsis

```cpp
class GStatic GWindowBase
{
    m_hWnd;
}
```

Base Classes

GWindowBase <-- GStatic

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- `Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)`
  Corresponds to `CStatic::Create`.
- `GetCursor ()`
  Corresponds to `CStatic::GetCursor`.
- `GetEnhMetaFile ()`
  Corresponds to `CStatic::GetEnhMetaFile`.
- `GetIcon ()`
  Corresponds to `CStatic::GetIcon`.
- `SetCursor (hCursor)`
  Corresponds to `CStatic::SetCursor`.
- `SetEnhMetaFile (hMetaFile)`
  Corresponds to `CStatic::SetEnhMetaFile`.
- `setIcon (hIcon)`
  Corresponds to `CStatic::SetIcon`.

This class also inherits the functions of GWindowBase.
GStatusBarCtrl

GStatusBarCtrl — an area at the bottom of a window that displays information.

Synopsis

class GStatusBarCtrl : GWindowBase
{
    m_hWnd;
}

Base Classes

GWindowBase <-- GStatusBarCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
    Corresponds to CStatusBarCtrl::AddControlResizeFlags.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CStatusBarCtrl::Create.
EndResizeGroup ()
    Corresponds to CStatusBarCtrl::EndResizeGroup.
GetBorders (args?...)
    Corresponds to CStatusBarCtrl::GetBorders.
GetIcon (nPane)
    Corresponds to CStatusBarCtrl::GetIcon.
GetRect (nPane, lpRect)
    Corresponds to CStatusBarCtrl::GetRect.
GetText (args?...)
    Corresponds to CStatusBarCtrl::GetText.
GetTextBSTR (args?...)
    Corresponds to CStatusBarCtrl::GetTextBSTR.
GetTextLength (args?...)
    Corresponds to CStatusBarCtrl::GetTextLength.
GetTipText (nPane, lpstrText, nSize)
    Corresponds to CStatusBarCtrl::GetTipText.
GetUnicodeFormat ()
Corresponds to CStatusBarCtrl::GetUnicodeFormat.

IsSimple ()
Corresponds to CStatusBarCtrl::IsSimple.

SetBkColor (clrBk)
Corresponds to CStatusBarCtrl::SetBkColor.

SetIcon (nPane, hIcon)
Corresponds to CStatusBarCtrl::SetIcon.

SetMinHeight (nMin)
Corresponds to CStatusBarCtrl::SetMinHeight.

SetSimple (args?)
Corresponds to CStatusBarCtrl::SetSimple.

SetText (args?)
Corresponds to CStatusBarCtrl::SetText.

SetTipText (nPane, lpstrText)
Corresponds to CStatusBarCtrl::SetTipText.

SetUnicodeFormat (args?)
Corresponds to CStatusBarCtrl::SetUnicodeFormat.

StartResizeGroup ()
Corresponds to CStatusBarCtrl::StartResizeGroup.

This class also inherits the functions of GWinboxBase.
**GTabCtrl**

GTabCtrl — a tabbed label for marking multiple pages.

**Synopsis**

```c
class GTabCtrl GWindowBase
{
    ChangeFunctions;
    _SequentialCtrlID;
    _children;
    m_hWnd;
}
```

**Base Classes**

GWindowBase <-- GTabCtrl

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **AddControlResizeFlags (ctrlid, flags)**
  - Corresponds to CTabCtrl::AddControlResizeFlags.

- **AddWindow (tabno, label, win)**
  - Corresponds to CTabCtrl::AddWindow.

- **AdjustRect (bLarger, lpRect)**
  - Corresponds to CTabCtrl::AdjustRect.

- **Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)**
  - Corresponds to CTabCtrl::Create.

- **DeleteAllItems ()**
  - Corresponds to CTabCtrl::DeleteAllItems.

- **DeleteItem (nItem)**
  - Corresponds to CTabCtrl::DeleteItem.

- **DeselectAll (args?...)**
  - Corresponds to CTabCtrl::DeselectAll.

- **EndResizeGroup ()**
  - Corresponds to CTabCtrl::EndResizeGroup.
GetCurFocus ()
    Corresponds to CTabCtrl::GetCurFocus.

GetCurSel ()
    Corresponds to CTabCtrl::GetCurSel.

GetExtendedStyle ()
    Corresponds to CTabCtrl::GetExtendedStyle.

GetItem (nItem, pTabCtrlItem)
    Corresponds to CTabCtrl::GetItem.

GetItemCount ()
    Corresponds to CTabCtrl::GetItemCount.

GetItemRect (nItem, lpRect)
    Corresponds to CTabCtrl::GetItemRect.

GetRowCount ()
    Corresponds to CTabCtrl::GetRowCount.

GetUnicodeFormat ()
    Corresponds to CTabCtrl::GetUnicodeFormat.

HighlightItem (args?...)
    Corresponds to CTabCtrl::HighlightItem.

HitTest (pHitTestInfo)
    Corresponds to CTabCtrl::HitTest.

InsertItem (nItem, pTabCtrlItem)
    Corresponds to CTabCtrl::InsertItem.

RemoveImage (nImage)
    Corresponds to CTabCtrl::RemoveImage.

SelChange ()
    Corresponds to CTabCtrl::SelChange.

SetCurFocus (nItem)
    Corresponds to CTabCtrl::SetCurFocus.

SetCurSel (nItem)
    Corresponds to CTabCtrl::SetCurSel.

SetExtendedStyle (dwExMask, dwExStyle)
    Corresponds to CTabCtrl::SetExtendedStyle.

SetItem (nItem, pTabCtrlItem)
    Corresponds to CTabCtrl::SetItem.

SetItemExtra (cbExtra)
    Corresponds to CTabCtrl::SetItemExtra.

SetItemSize (size)
    Corresponds to CTabCtrl::SetItemSize.
SetMinTabWidth (args?...)  
Corresponds to CTabCtrl::SetMinTabWidth.

SetPadding (size)  
Corresponds to CTabCtrl::SetPadding.

SetToolTips (hWndToolTip)  
Corresponds to CTabCtrl::SetToolTips.

SetUnicodeFormat (args?...)  
Corresponds to CTabCtrl::SetUnicodeFormat.

StartResizeGroup ()  
Corresponds to CTabCtrl::StartResizeGroup.

This class also inherits the functions of GWindowBase.
GToolBarCtrl

GToolBarCtrl — a toolbar with one or more buttons.

Synopsis

```cpp
class GToolBarCtrl GWindowBase
{
    m_hWnd;
}
```

Base Classes

GWindowBase <-- GToolBarCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **AddBitmap (args?)**: Corresponds to CToolBarCtrl::AddBitmap.
- **AddButtons (nNumButtons, lpButtons)**: Corresponds to CToolBarCtrl::AddButtons.
- **AddControlResizeFlags (ctrlid, flags)**: Corresponds to CToolBarCtrl::AddControlResizeFlags.
- **AddString (nStringID)**: Corresponds to CToolBarCtrl::AddString.
- **AddStrings (lpszStrings)**: Corresponds to CToolBarCtrl::AddStrings.
- **AutoSize ()**: Corresponds to CToolBarCtrl::AutoSize.
- **ChangeBitmap (nID, nBitmap)**: Corresponds to CToolBarCtrl::ChangeBitmap.
- **CheckButton (args?)**: Corresponds to CToolBarCtrl::CheckButton.
- **CommandToIndex (nID)**: Corresponds to CToolBarCtrl::CommandToIndex.
- **Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)**: Corresponds to CToolBarCtrl::Create.
Customize ()
Corresponds to CToolBarCtrl::Customize.

DeleteButton (nIndex)
Corresponds to CToolBarCtrl::DeleteButton.

EnableButton (args?...)
Corresponds to CToolBarCtrl::EnableButton.

EndResizeGroup ()
Corresponds to CToolBarCtrl::EndResizeGroup.

GetAnchorHighlight ()
Corresponds to CToolBarCtrl::GetAnchorHighlight.

GetBitmap (nID)
Corresponds to CToolBarCtrl::GetBitmap.

GetBitmapFlags ()
Corresponds to CToolBarCtrl::GetBitmapFlags.

GetButton (nIndex, lpButton)
Corresponds to CToolBarCtrl::GetButton.

GetButtonCount ()
Corresponds to CToolBarCtrl::GetButtonCount.

GetButtonInfo (nID, lptbbi)
Corresponds to CToolBarCtrl::GetButtonInfo.

GetButtonSize (args?...)
Corresponds to CToolBarCtrl::GetButtonSize.

GetButtonText (nID, lpstrText)
Corresponds to CToolBarCtrl::GetButtonText.

GetColorScheme (lpcs)
Corresponds to CToolBarCtrl::GetColorScheme.

GetExtendedStyle ()
Corresponds to CToolBarCtrl::GetExtendedStyle.

GetHotItem ()
Corresponds to CToolBarCtrl::GetHotItem.

GetInsertMarkColor ()
Corresponds to CToolBarCtrl::GetInsertMarkColor.

GetItemRect (nIndex, lpRect)
Corresponds to CToolBarCtrl::GetItemRect.

GetMaxSize (lpSize)
Corresponds to CToolBarCtrl::GetMaxSize.

GetPadding (lpSizePadding)
Corresponds to CToolBarCtrl::GetPadding.
GetRect (nID, lpRect)
   Corresponds to CToolBarCtrl::GetRect.

GetRows ()
   Corresponds to CToolBarCtrl::GetRows.

GetState (nID)
   Corresponds to CToolBarCtrl::GetState.

GetString (nString, lpstrString, cchMaxLen)
   Corresponds to CToolBarCtrl::GetString.

GetStyle ()
   Corresponds to CToolBarCtrl::GetStyle.

GetTextRows ()
   Corresponds to CToolBarCtrl::GetTextRows.

GetUnicodeFormat ()
   Corresponds to CToolBarCtrl::GetUnicodeFormat.

HideButton (args?...)
   Corresponds to CToolBarCtrl::HideButton.

HitTest (lpPoint)
   Corresponds to CToolBarCtrl::HitTest.

Indeterminate (args?...)
   Corresponds to CToolBarCtrl::Indeterminate.

InsertButton (nIndex, lpButton)
   Corresponds to CToolBarCtrl::InsertButton.

InsertMarkHitTest (args?...)
   Corresponds to CToolBarCtrl::InsertMarkHitTest.

IsButtonChecked (nID)
   Corresponds to CToolBarCtrl::IsButtonChecked.

IsButtonEnabled (nID)
   Corresponds to CToolBarCtrl::IsButtonEnabled.

IsButtonHidden (nID)
   Corresponds to CToolBarCtrl::IsButtonHidden.

IsButtonHighlighted (nButtonID)
   Corresponds to CToolBarCtrl::IsButtonHighlighted.

IsButtonIndeterminate (nID)
   Corresponds to CToolBarCtrl::IsButtonIndeterminate.

IsButtonPressed (nID)
   Corresponds to CToolBarCtrl::IsButtonPressed.

LoadImages (nBitmapID)
   Corresponds to CToolBarCtrl::LoadImages.
LoadStdImages (nBitmapID)
  Corresponds to CToolBarCtrl::LoadStdImages.
MarkButton (args?...)
  Corresponds to CToolBarCtrl::MarkButton.
MoveButton (nOldPos, nNewPos)
  Corresponds to CToolBarCtrl::MoveButton.
PressButton (args?...)
  Corresponds to CToolBarCtrl::PressButton.
RestoreState (hKeyRoot, lpszSubKey, lpszValueName)
  Corresponds to CToolBarCtrl::RestoreState.
SaveState (hKeyRoot, lpszSubKey, lpszValueName)
  Corresponds to CToolBarCtrl::SaveState.
SetAnchorHighlight (args?...)
  Corresponds to CToolBarCtrl::SetAnchorHighlight.
SetBitmapSize (args?...)
  Corresponds to CToolBarCtrl::SetBitmapSize.
SetButtonInfo (nID, lptbbi)
  Corresponds to CToolBarCtrl::SetButtonInfo.
SetButtonSize (args?...)
  Corresponds to CToolBarCtrl::SetButtonSize.
SetButtonStructSize (args?...)
  Corresponds to CToolBarCtrl::SetButtonStructSize.
SetButtonWidth (cxMin, cxMax)
  Corresponds to CToolBarCtrl::SetButtonWidth.
SetCmdID (nIndex, nID)
  Corresponds to CToolBarCtrl::SetCmdID.
SetColorScheme (lpcs)
  Corresponds to CToolBarCtrl::SetColorScheme.
SetDrawTextFlags (dwMask, dwFlags)
  Corresponds to CToolBarCtrl::SetDrawTextFlags.
SetExtendedStyle (dwStyle)
  Corresponds to CToolBarCtrl::SetExtendedStyle.
SetIndent (nIndent)
  Corresponds to CToolBarCtrl::SetIndent.
SetInsertMarkColor (clr)
  Corresponds to CToolBarCtrl::SetInsertMarkColor.
SetMaxTextRows (nMaxTextRows)
    Corresponds to CToolBarCtrl::SetMaxTextRows.
SetNotifyWnd (hWnd)
    Corresponds to CToolBarCtrl::SetNotifyWnd.
SetPadding (args?...)
    Corresponds to CToolBarCtrl::SetPadding.
SetRows (nRows, bLarger, lpRect)
    Corresponds to CToolBarCtrl::SetRows.
SetState (nID, nState)
    Corresponds to CToolBarCtrl::SetState.
SetStyle (dwStyle)
    Corresponds to CToolBarCtrl::SetStyle.
SetToolTip (hWndToolTip)
    Corresponds to CToolBarCtrl::SetToolTip.
SetUnicodeFormat (args?...)
    Corresponds to CToolBarCtrl::SetUnicodeFormat.
StartResizeGroup ()
    Corresponds to CToolBarCtrl::StartResizeGroup.

This class also inherits the functions of GWindowBase.
**GToolTipCtrl**

GToolTipCtrl — a pop-up window with a line of text.

**Synopsis**

```cpp
class GToolTipCtrl GWindowBase
{
    m_hWnd;
}
```

**Base Classes**

GWindowBase <= GToolTipCtrl

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **Activate (bActivate)**
  - Corresponds to CToolTipCtrl::Activate.
- **AddControlResizeFlags (ctrlid, flags)**
  - Corresponds to CToolTipCtrl::AddControlResizeFlags.
- **AdjustRect (lpRect, bLarger)**
  - Corresponds to CToolTipCtrl::AdjustRect.
- **Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)**
  - Corresponds to CToolTipCtrl::Create.
- **DelTool (args?...)**
  - Corresponds to CToolTipCtrl::DelTool.
- **EndResizeGroup ()**
  - Corresponds to CToolTipCtrl::EndResizeGroup.
- **EnumTools (nTool, lpToolInfo)**
  - Corresponds to CToolTipCtrl::EnumTools.
- **GetBubbleSize (lpToolInfo)**
  - Corresponds to CToolTipCtrl::GetBubbleSize.
- **GetCurrentTool (lpToolInfo)**
  - Corresponds to CToolTipCtrl::GetCurrentTool.
- **GetDelayTime (dwType)**
  - Corresponds to CToolTipCtrl::GetDelayTime.
GetMargin (lpRect)
   Corresponds to CToolTipCtrl::GetMargin.

GetMaxTipWidth ()
   Corresponds to CToolTipCtrl::GetMaxTipWidth.

GetText (args?...)
   Corresponds to CToolTipCtrl::GetText.

GetTipBkColor ()
   Corresponds to CToolTipCtrl::GetTipBkColor.

GetTipTextColor ()
   Corresponds to CToolTipCtrl::GetTipTextColor.

GetTitle (pTTGetTitle)
   Corresponds to CToolTipCtrl::GetTitle.

GetToolCount ()
   Corresponds to CToolTipCtrl::GetToolCount.

GetToolInfo (args?...)
   Corresponds to CToolTipCtrl::GetToolInfo.

HitTest (args?...)
   Corresponds to CToolTipCtrl::HitTest.

Pop ()
   Corresponds to CToolTipCtrl::Pop.

Popup ()
   Corresponds to CToolTipCtrl::Popup.

RelayEvent (lpMsg)
   Corresponds to CToolTipCtrl::RelayEvent.

SetDelayTime (dwType, nTime)
   Corresponds to CToolTipCtrl::SetDelayTime.

SetMargin (lpRect)
   Corresponds to CToolTipCtrl::SetMargin.

SetMaxTipWidth (nWidth)
   Corresponds to CToolTipCtrl::SetMaxTipWidth.

SetTipBkColor (clr)
   Corresponds to CToolTipCtrl::SetTipBkColor.

SetTipTextColor (clr)
   Corresponds to CToolTipCtrl::SetTipTextColor.

SetTitle (uIcon, lpstrTitle)
   Corresponds to CToolTipCtrl::SetTitle.

SetToolInfo (lpToolInfo)
   Corresponds to CToolTipCtrl::SetToolInfo.
SetToolRect (args?)
  
  Corresponds to CToolTipCtrl::SetToolRect.

StartResizeGroup ()
  
  Corresponds to CToolTipCtrl::StartResizeGroup.

TrackActivate (lpToolInfo, bActivate)
  
  Corresponds to CToolTipCtrl::TrackActivate.

TrackPosition (xPos, yPos)
  
  Corresponds to CToolTipCtrl::TrackPosition.

Update ()
  
  Corresponds to CToolTipCtrl::Update.

This class also inherits the functions of GWindowBase.
GTrackBarCtrl

GTrackBarCtrl — a slider with optional tick marks, based on TrackBar Control.

Synopsis

class GTrackBarCtrl : GWindowBase
{
    m_hWnd;
}

Base Classes

GWindowBase <- GTrackBarCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
  Corresponds to CTrackBarCtrl::AddControlResizeFlags.
ClearSel (args?...)
  Corresponds to CTrackBarCtrl::ClearSel.
ClearTics (args?...)
  Corresponds to CTrackBarCtrl::ClearTics.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
  Corresponds to CTrackBarCtrl::Create.
EndResizeGroup ()
  Corresponds to CTrackBarCtrl::EndResizeGroup.
GetBuddy (args?...)
  Corresponds to CTrackBarCtrl::GetBuddy.
GetChannelRect (lprc)
  Corresponds to CTrackBarCtrl::GetChannelRect.
GetLineSize ()
  Corresponds to CTrackBarCtrl::GetLineSize.
GetNumTics ()
  Corresponds to CTrackBarCtrl::GetNumTics.
GetPageSize ()
  Corresponds to CTrackBarCtrl::GetPageSize.
GetPosition ()
Corresponds to CTrackBarCtrl::GetPosition.

GetRangeMax ()
Corresponds to CTrackBarCtrl::GetRangeMax.

GetRangeMin ()
Corresponds to CTrackBarCtrl::GetRangeMin.

GetSelEnd ()
Corresponds to CTrackBarCtrl::GetSelEnd.

GetSelStart ()
Corresponds to CTrackBarCtrl::GetSelStart.

GetThumbLength ()
Corresponds to CTrackBarCtrl::GetThumbLength.

GetThumbRect (lprc)
Corresponds to CTrackBarCtrl::GetThumbRect.

GetTic (nTic)
Corresponds to CTrackBarCtrl::GetTic.

GetTicPos (nTic)
Corresponds to CTrackBarCtrl::GetTicPos.

GetUnicodeFormat ()
Corresponds to CTrackBarCtrl::GetUnicodeFormat.

SetBuddy (args?)
Corresponds to CTrackBarCtrl::SetBuddy.

SetLineSize (nSize)
Corresponds to CTrackBarCtrl::SetLineSize.

SetPageSize (nSize)
Corresponds to CTrackBarCtrl::SetPageSize.

SetPos (nPos)
Corresponds to CTrackBarCtrl::SetPos.

SetRange (args?)
Corresponds to CTrackBarCtrl::SetRange.

SetRangeMax (args?)
Corresponds to CTrackBarCtrl::SetRangeMax.

SetRangeMin (args?)
Corresponds to CTrackBarCtrl::SetRangeMin.

SetSel (args?)
Corresponds to CTrackBarCtrl::SetSel.

SetSelEnd (nMax)
Corresponds to CTrackBarCtrl::SetSelEnd.
SetSelStart (nMin)
   Corresponds to CTrackBarCtrl::SetSelStart.
SetSelection (nMin, nMax)
   Corresponds to CTrackBarCtrl::SetSelection.
SetThumbLength (nLength)
   Corresponds to CTrackBarCtrl::SetThumbLength.
SetTic (nTic)
   Corresponds to CTrackBarCtrl::SetTic.
SetTicFreq (nFreq)
   Corresponds to CTrackBarCtrl::SetTicFreq.
SetTipSide (nSide)
   Corresponds to CTrackBarCtrl::SetTipSide.
SetToolTips (hWndTT)
   Corresponds to CTrackBarCtrl::SetToolTips.
SetUnicodeFormat (args?...)
   Corresponds to CTrackBarCtrl::SetUnicodeFormat.
StartResizeGroup ()
   Corresponds to CTrackBarCtrl::StartResizeGroup.
VerifyPos ()
   Corresponds to CTrackBarCtrl::VerifyPos.

This class also inherits the functions of GWindowBase.
**GTreeViewCtrl**

GTreeViewCtrl — displays a hierarchy of nodes as a tree, based on TreeView Control.

**Synopsis**

```cpp
class GTreeViewCtrl GWindowBase
{
    m_hWnd;
}
```

**Base Classes**

GWindowBase <-- GTreeViewCtrl

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
   Corresponds to CTreeViewCtrl::AddControlResizeFlags.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
   Corresponds to CTreeViewCtrl::Create.
DeleteAllItems ()
   Corresponds to CTreeViewCtrl::DeleteAllItems.
DeleteItem (hItem)
   Corresponds to CTreeViewCtrl::DeleteItem.
EndEditLabelNow (bCancel)
   Corresponds to CTreeViewCtrl::EndEditLabelNow.
EndResizeGroup ()
   Corresponds to CTreeViewCtrl::EndResizeGroup.
EnsureVisible (hItem)
   Corresponds to CTreeViewCtrl::EnsureVisible.
Expand (args?...)
   Corresponds to CTreeViewCtrl::Expand.
GetBkColor ()
   Corresponds to CTreeViewCtrl::GetBkColor.
GetCheckState (hItem)
Corresponds to CTreeViewCtrl::GetCheckState.

GetChildItem (hItem)
Corresponds to CTreeViewCtrl::GetChildItem.

GetCount ()
Corresponds to CTreeViewCtrl::GetCount.

GetDropHighlightItem ()
Corresponds to CTreeViewCtrl::GetDropHighlightItem.

GetFirstVisibleItem ()
Corresponds to CTreeViewCtrl::GetFirstVisibleItem.

GetISearchString (lpstr)
Corresponds to CTreeViewCtrl::GetISearchString.

Indent ()
Corresponds to CTreeViewCtrl::GetIndent.

GetInsertMarkColor ()
Corresponds to CTreeViewCtrl::GetInsertMarkColor.

GetItem (args?)
Corresponds to CTreeViewCtrl::GetItem.

GetItemData (hItem)
Corresponds to CTreeViewCtrl::GetItemData.

GetItemHeight ()
Corresponds to CTreeViewCtrl::GetItemHeight.

GetItemRect (hItem, lpRect, bTextOnly)
Corresponds to CTreeViewCtrl::GetItemRect.

GetItemState (hItem, nStateMask)
Corresponds to CTreeViewCtrl::GetItemState.

GetItemText (args?)
Corresponds to CTreeViewCtrl::GetItemText.

GetLineColor ()
Corresponds to CTreeViewCtrl::GetLineColor.

GetNextItem (hItem, nCode)
Corresponds to CTreeViewCtrl::GetNextItem.

GetNextSiblingItem (hItem)
Corresponds to CTreeViewCtrl::GetNextSiblingItem.

GetNextVisibleItem (hItem)
Corresponds to CTreeViewCtrl::GetNextVisibleItem.

GetParentItem (hItem)
Corresponds to CTreeViewCtrl::GetParentItem.
GetPrevSiblingItem (hItem)
Corresponds to CTreeViewCtrl::GetPrevSiblingItem.

GetPrevVisibleItem (hItem)
Corresponds to CTreeViewCtrl::GetPrevVisibleItem.

GetRootItem ()
Corresponds to CTreeViewCtrl::GetRootItem.

GetScrollTime ()
Corresponds to CTreeViewCtrl::GetScrollTime.

GetSelectedItem ()
Corresponds to CTreeViewCtrl::GetSelectedItem.

GetTextColor ()
Corresponds to CTreeViewCtrl::GetTextColor.

GetUnicodeFormat ()
Corresponds to CTreeViewCtrl::GetUnicodeFormat.

GetVisibleCount ()
Corresponds to CTreeViewCtrl::GetVisibleCount.

HitTest (args?...)
Corresponds to CTreeViewCtrl::HitTest.

InsertItem (args?...)
Corresponds to CTreeViewCtrl::InsertItem.

ItemHasChildren (hItem)
Corresponds to CTreeViewCtrl::ItemHasChildren.

MapAccIDToHTREEITEM (uID)
Corresponds to CTreeViewCtrl::MapAccIDToHTREEITEM.

MapHTREEITEMToAccID (hTreeItem)
Corresponds to CTreeViewCtrl::MapHTREEITEMToAccID.

RemoveInsertMark ()
Corresponds to CTreeViewCtrl::RemoveInsertMark.

Select (hItem, nCode)
Corresponds to CTreeViewCtrl::Select.

SelectDropTarget (hItem)
Corresponds to CTreeViewCtrl::SelectDropTarget.

SelectItem (hItem)
Corresponds to CTreeViewCtrl::SelectItem.

SelectSetFirstVisible (hItem)
Corresponds to CTreeViewCtrl::SelectSetFirstVisible.

SetBkColor (clr)
Corresponds to CTreeViewCtrl::SetBkColor.
Widgets

SetCheckState (hItem, bCheck)
    Corresponds to CTreeViewCtrl::SetCheckState.

SetIndent (nIndent)
    Corresponds to CTreeViewCtrl::SetIndent.

SetInsertMark (hTreeItem, bAfter)
    Corresponds to CTreeViewCtrl::SetInsertMark.

SetInsertMarkColor (clr)
    Corresponds to CTreeViewCtrl::SetInsertMarkColor.

SetItem (args?...)
    Corresponds to CTreeViewCtrl::SetItem.

SetItemData (hItem, dwData)
    Corresponds to CTreeViewCtrl::SetItemData.

SetItemHeight (cyHeight)
    Corresponds to CTreeViewCtrl::SetItemHeight.

SetItemImage (hItem, nImage, nSelectedImage)
    Corresponds to CTreeViewCtrl::SetItemImage.

SetItemState (hItem, nState, nStateMask)
    Corresponds to CTreeViewCtrl::SetItemState.

SetItemText (hItem, lpszItem)
    Corresponds to CTreeViewCtrl::SetItemText.

SetLineColor (clrNew)
    Corresponds to CTreeViewCtrl::SetLineColor.

SetScrollTime (nScrollTime)
    Corresponds to CTreeViewCtrl::SetScrollTime.

SetTextColor (clr)
    Corresponds to CTreeViewCtrl::SetTextColor.

SetUnicodeFormat (args?...)
    Corresponds to CTreeViewCtrl::SetUnicodeFormat.

SortChildren (args?...)
    Corresponds to CTreeViewCtrl::SortChildren.

SortChildrenCB (args?...)
    Corresponds to CTreeViewCtrl::SortChildrenCB.

StartResizeGroup ()
    Corresponds to CTreeViewCtrl::StartResizeGroup.

This class also inherits the functions of GW根底Base.
**GTreeViewCtrlEx**

GTreeViewCtrlEx — an extended TreeViewCtrl.

**Synopsis**

```
class GTreeViewCtrlEx GWindowBase
{
    m_hWnd;
}
```

**Base Classes**

`GWindowBase <--> GTreeViewCtrlEx`

**Class Members**

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **AddControlResizeFlags (ctrlid, flags)**  
  Corresponds to `CTreeViewCtrlEx::AddControlResizeFlags`.

- **Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)**  
  Corresponds to `CTreeViewCtrlEx::Create`.

- **EndResizeGroup ()**  
  Corresponds to `CTreeViewCtrlEx::EndResizeGroup`.

- **StartResizeGroup ()**  
  Corresponds to `CTreeViewCtrlEx::StartResizeGroup`.

This class also inherits the functions of `GWindowBase`. 
GUpDownCtrl

GUpDownCtrl — a pair of arrows for incrementing and decrementing values, based on UpDown Control.

Synopsis

class GUpDownCtrl GWindowBase
{
   m_hWnd;
}

Base Classes

GWindowBase <-- GUpDownCtrl

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
   Corresponds to CUpDownCtrl::Create.
GetAccel (nAccel, pAccel)
   Corresponds to CUpDownCtrl::GetAccel.
GetBase ()
   Corresponds to CUpDownCtrl::GetBase.
GetPos (args?...)
   Corresponds to CUpDownCtrl::GetPos.
GetPos32 (args?...)
   Corresponds to CUpDownCtrl::GetPos32.
GetRange (args?...)
   Corresponds to CUpDownCtrl::GetRange.
GetUnicodeFormat ()
   Corresponds to CUpDownCtrl::GetUnicodeFormat.
SetAccel (nAccel, pAccel)
   Corresponds to CUpDownCtrl::SetAccel.
SetBase (nBase)
   Corresponds to CUpDownCtrl::SetBase.
SetPos (nPos)
  Corresponds to CUpDownCtrl::SetPos.
SetPos32 (nPos)
  Corresponds to CUpDownCtrl::SetPos32.
SetRange (nLower, nUpper)
  Corresponds to CUpDownCtrl::SetRange.
SetRange32 (nLower, nUpper)
  Corresponds to CUpDownCtrl::SetRange32.
SetUnicodeFormat (args?)
  Corresponds to CUpDownCtrl::SetUnicodeFormat.

This class also inherits the functions of GWWindowBase.
GWWindow

GWWindow — methods for manipulating a window.

Synopsis

class GWWindow GWWindowBase
{
    _m_hWnd;
}

Base Classes

GWWindowBase <-- GWWindow

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

AddControlResizeFlags (ctrlid, flags)
    Corresponds to CWindow::AddControlResizeFlags.
Create (hWndParent, rect, szWindowName, dwStyle, dwExStyle)
    Corresponds to CWindow::Create.
EndResizeGroup ()
    Corresponds to CWindow::EndResizeGroup.
StartResizeGroup ()
    Corresponds to CWindow::StartResizeGroup.

This class also inherits the functions of GWWindowBase.
GWIndowBase

GWIndowBase — the base for most of these widgets, based on CWindow.

Synopsis

```cpp
class GWindowBase
{
    ChangeFunctions;
    _SequentialCtrlID;
    m_hWnd;
}
```

Class Members

These functions correspond to Windows functions, but may not have exactly the same name. The links attempt a search on the name, but you might have to refine the search manually.

- **ArrangeIconicWindows ()**
  `Corresponds to CWindowBase::ArrangeIconicWindows.`

- **Attach (hWndNew)**
  `Corresponds to CWindowBase::Attach.`

- **BeginPaint (lpPaint)**
  `Corresponds to CWindowBase::BeginPaint.`

- **BringWindowToTop ()**
  `Corresponds to CWindowBase::BringWindowToTop.`

- **CenterWindow (args?...)**
  `Corresponds to CWindowBase::CenterWindow.`

- **ChangeClipboardChain (hWndNewNext)**
  `Corresponds to CWindowBase::ChangeClipboardChain.`

- **CheckDlgButton (nIDButton, nCheck)**
  `Corresponds to CWindowBase::CheckDlgButton.`

- **CheckRadioButton (nIDFirstButton, nIDLastButton, nIDCheckButton)**
  `Corresponds to CWindowBase::CheckRadioButton.`

- **ClientToScreen (args?...)**
  `Corresponds to CWindowBase::ClientToScreen.`

- **CommandHandler (wNotifyCode, wID, code)**
  `Corresponds to CWindowBase::CommandHandler.```
Widgets

CreateCaret (hBitmap)
Corresponds to CWindowBase::CreateCaret.

CreateControl (klass, x, y, w, h, label, style_add?=0, exstyle_add=0, style_del=0, exstyle_del=0)
Corresponds to CWindowBase::CreateControl.

CreateGrayCaret (nWidth, nHeight)
Corresponds to CWindowBase::CreateGrayCaret.

CreateSolidCaret (nWidth, nHeight)
Corresponds to CWindowBase::CreateSolidCaret.

DeferWindowPos (hWinPosInfo, hWndInsertAfter, x, y, cx, cy, uFlags)
Corresponds to CWindowBase::DeferWindowPos.

DestroyWindow ()
Corresponds to CWindowBase::DestroyWindow.

Detach ()
Corresponds to CWindowBase::Detach.

DlgDirList (lpPathSpec, nIDListBox, nIDStaticPath, nFileType)
Corresponds to CWindowBase::DlgDirList.

DlgDirListComboBox (lpPathSpec, nIDComboBox, nIDStaticPath, nFileType)
Corresponds to CWindowBase::DlgDirListComboBox.

DlgDirSelect (lpString, nCount, nIDListBox)
Corresponds to CWindowBase::DlgDirSelect.

DlgDirSelectComboBox (lpString, nCount, nIDComboBox)
Corresponds to CWindowBase::DlgDirSelectComboBox.

DragAcceptFiles (args?...)
Corresponds to CWindowBase::DragAcceptFiles.

DrawMenuBar ()
Corresponds to CWindowBase::DrawMenuBar.

EnableScrollBar (args?...)
Corresponds to CWindowBase::EnableScrollBar.

EnableWindow (args?...)
Corresponds to CWindowBase::EnableWindow.

EndPaint (lpPaint)
Corresponds to CWindowBase::EndPaint.

FlashWindow (bInvert)
Corresponds to CWindowBase::FlashWindow.

GetClientRect (lpRect)
Corresponds to CWindowBase::GetClientRect.
GetDC ()
    Corresponds to CWindowBase::GetDC.
GetDCEx (hRgnClip, flags)
    Corresponds to CWindowBase::GetDCEx.
GetDlgItemID ()
    Corresponds to CWindowBase::GetDlgItemID.
GetDlgItemInt (args?...)
    Corresponds to CWindowBase::GetDlgItemInt.
GetDlgItemText (args?...)
    Corresponds to CWindowBase::GetDlgItemText.
GetExStyle ()
    Corresponds to CWindowBase::GetExStyle.
GetFont ()
    Corresponds to CWindowBase::GetFont.
GetHotKey ()
    Corresponds to CWindowBase::GetHotKey.
GetHwnd ()
    Corresponds to CWindowBase::GetHwnd.
GetIcon (args?...)
    Corresponds to CWindowBase::GetIcon.
GetMenu ()
    Corresponds to CWindowBase::GetMenu.
GetNextDlgGroupItem (args?...)
    Corresponds to CWindowBase::GetNextDlgGroupItem.
GetNextDlgTabItem (args?...)
    Corresponds to CWindowBase::GetNextDlgTabItem.
GetScrollInfo (nBar, lpScrollInfo)
    Corresponds to CWindowBase::GetScrollInfo.
GetScrollPos (nBar)
    Corresponds to CWindowBase::GetScrollPos.
GetStyle ()
    Corresponds to CWindowBase::GetStyle.
GetSystemMenu (bRevert)
    Corresponds to CWindowBase::GetSystemMenu.
GetUpdateRect (args?...)
    Corresponds to CWindowBase::GetUpdateRect.
GetUpdateRgn (args?...)
    Corresponds to CWindowBase::GetUpdateRgn.
GetWindowContextHelpId ()
    Corresponds to CWindowBase::GetWindowContextHelpId.

GetWindowDC ()
    Corresponds to CWindowBase::GetWindowDC.

GetWindowLong (nIndex)
    Corresponds to CWindowBase::GetWindowLong.

GetWindowPlacement (lpwndpl)
    Corresponds to CWindowBase::GetWindowPlacement.

GetWindowProcessID ()
    Corresponds to CWindowBase::GetWindowProcessID.

GetWindowRect (lpRect)
    Corresponds to CWindowBase::GetWindowRect.

GetWindowRgn (hRgn)
    Corresponds to CWindowBase::GetWindowRgn.

GetWindowText ()
    Corresponds to CWindowBase::GetWindowText.

GetWindowTextLength ()
    Corresponds to CWindowBase::GetWindowTextLength.

GetWindowThreadID ()
    Corresponds to CWindowBase::GetWindowThreadID.

GetWindowWord (nIndex)
    Corresponds to CWindowBase::GetWindowWord.

GotoDlgCtrl (hWndCtrl)
    Corresponds to CWindowBase::GotoDlgCtrl.

HideCaret ()
    Corresponds to CWindowBase::HideCaret.

HiliteMenuItem (hMenu, uItemHilite, uHilite)
    Corresponds to CWindowBase::HiliteMenuItem.

ImplementedControl (klass, x, y, w, h, label)
    Corresponds to CWindowBase::ImplementedControl.

Invalidate (args?...)
    Corresponds to CWindowBase::Invalidate.

InvalidateRect (args?...)
    Corresponds to CWindowBase::InvalidateRect.

InvalidateRgn (args?...)
    Corresponds to CWindowBase::InvalidateRgn.

IsChild (hWnd)
    Corresponds to CWindowBase::IsChild.
IsDialogMessage (lpMsg)
    Corresponds to CWindowBase::IsDialogMessage.
IsDlgButtonChecked (nIDButton)
    Corresponds to CWindowBase::IsDlgButtonChecked.
IsIconic ()
    Corresponds to CWindowBase::IsIconic.
IsParentDialog ()
    Corresponds to CWindowBase::IsParentDialog.
IsWindow ()
    Corresponds to CWindowBase::IsWindow.
IsWindowEnabled ()
    Corresponds to CWindowBase::IsWindowEnabled.
IsWindowUnicode ()
    Corresponds to CWindowBase::IsWindowUnicode.
IsWindowVisible ()
    Corresponds to CWindowBase::IsWindowVisible.
IsZoomed ()
    Corresponds to CWindowBase::IsZoomed.
KillTimer (nIDEvent)
    Corresponds to CWindowBase::KillTimer.
LockWindowUpdate (args?...)
    Corresponds to CWindowBase::LockWindowUpdate.
MapWindowPoints (args?...)
    Corresponds to CWindowBase::MapWindowPoints.
MessageBox (args?...)
    Corresponds to CWindowBase::MessageBox.
MessageHandler (uMsg, code)
    Corresponds to CWindowBase::MessageHandler.
ModifyStyle (args?...)
    Corresponds to CWindowBase::ModifyStyle.
ModifyStyleEx (args?...)
    Corresponds to CWindowBase::ModifyStyleEx.
MoveWindow (args?...)
    Corresponds to CWindowBase::MoveWindow.
NextCtrlID ()
    Corresponds to CWindowBase::NextCtrlID.
NextDlgCtrl ()
    Corresponds to CWindowBase::NextDlgCtrl.
NotifyHandler (idCtrl, iNotifyCode, code)  
**Corresponds to** CWindowBase::NotifyHandler.

OpenClipboard ()  
**Corresponds to** CWindowBase::OpenClipboard.

PostMessage (args?)...  
**Corresponds to** CWindowBase::PostMessage.

PrevDlgCtrl ()  
**Corresponds to** CWindowBase::PrevDlgCtrl.

Print (hDC, dwFlags)  
**Corresponds to** CWindowBase::Print.

PrintClient (hDC, dwFlags)  
**Corresponds to** CWindowBase::PrintClient.

RedrawWindow (args?)...  
**Corresponds to** CWindowBase::RedrawWindow.

ReleaseDC (hDC)  
**Corresponds to** CWindowBase::ReleaseDC.

ResizeClient (args?)...  
**Corresponds to** CWindowBase::ResizeClient.

ScreenToClient (args?)...  
**Corresponds to** CWindowBase::ScreenToClient.

ScrollWindow (args?)...  
**Corresponds to** CWindowBase::ScrollWindow.

ScrollWindowEx (args?)...  
**Corresponds to** CWindowBase::ScrollWindowEx.

SendMessageToDescendants (args?)...  
**Corresponds to** CWindowBase::SendMessageToDescendants.

SetActiveWindow ()  
**Corresponds to** CWindowBase::SetActiveWindow.

SetCapture ()  
**Corresponds to** CWindowBase::SetCapture.

SetBackground (style, color, hatch)  
**Corresponds to** CWindowBase::SetBackground.
SetChildBackground (style, color, hatch)
  \hspace{1em} Corresponds to CWindowBase::SetChildBackground.

SetChildFontEx (nHeight, nWidth, nEscapement, nOrientation, nWeight, 
bitalic, bUnderline, cStrikeOut, nCharSet, nOutPrecision, nClipPrecision, 
nQuality, nPitchAndFamily, lpszFacename)
  \hspace{1em} Corresponds to CWindowBase::SetChildFontEx.

SetChildForeground (penstyles, width, color)
  \hspace{1em} Corresponds to CWindowBase::SetChildForeground.

SetClipboardViewer ()
  \hspace{1em} Corresponds to CWindowBase::SetClipboardViewer.

SetDlgCtrlID (nID)
  \hspace{1em} Corresponds to CWindowBase::SetDlgCtrlID.

SetDlgItemInt (args?...)
  \hspace{1em} Corresponds to CWindowBase::SetDlgItemInt.

SetDlgItemText (nID, lpszString)
  \hspace{1em} Corresponds to CWindowBase::SetDlgItemText.

SetFocus ()
  \hspace{1em} Corresponds to CWindowBase::SetFocus.

SetFont (args?...)
  \hspace{1em} Corresponds to CWindowBase::SetFont.

SetFontEx (nHeight, nWidth, nEscapement, nOrientation, nWeight, 
bitalic, bUnderline, cStrikeOut, nCharSet, nOutPrecision, nClipPrecision, 
nQuality, nPitchAndFamily, lpszFacename)
  \hspace{1em} Corresponds to CWindowBase::SetFontEx.

SetForeground (penstyles, width, color)
  \hspace{1em} Corresponds to CWindowBase::SetForeground.

SetHotKey (wVirtualKeyCode, wModifiers)
  \hspace{1em} Corresponds to CWindowBase::SetHotKey.

SetIcon (args?...)
  \hspace{1em} Corresponds to CWindowBase::SetIcon.

SetMenu (hMenu)
  \hspace{1em} Corresponds to CWindowBase::SetMenu.

SetRedraw (args?...)
  \hspace{1em} Corresponds to CWindowBase::SetRedraw.

SetScrollInfo (args?...)
  \hspace{1em} Corresponds to CWindowBase::SetScrollInfo.

SetScrollPos (args?...)
  \hspace{1em} Corresponds to CWindowBase::SetScrollPos.
SetScrollRange (args?)
  Corresponds to CWindowBase::SetScrollRange.

SetTimer (args?)
  Corresponds to CWindowBase::SetTimer.

SetWindowContextHelpId (dwContextHelpId)
  Corresponds to CWindowBase::SetWindowContextHelpId.

SetWindowLong (args?)
  Corresponds to CWindowBase::SetWindowLong.

SetWindowPlacement (lpwndpl)
  Corresponds to CWindowBase::SetWindowPlacement.

SetWindowPos (args?)
  Corresponds to CWindowBase::SetWindowPos.

SetWindowRgn (args?)
  Corresponds to CWindowBase::SetWindowRgn.

SetWindowText (lpszString)
  Corresponds to CWindowBase::SetWindowText.

SetWindowWord (nIndex, wNewWord)
  Corresponds to CWindowBase::SetWindowWord.

ShowCaret ()
  Corresponds to CWindowBase::ShowCaret.

ShowOwnedPopups (args?)
  Corresponds to CWindowBase::ShowOwnedPopups.

ShowScrollBar (args?)
  Corresponds to CWindowBase::ShowScrollBar.

ShowWindow (nCmdShow)
  Corresponds to CWindowBase::ShowWindow.

ShowWindowAsync (nCmdShow)
  Corresponds to CWindowBase::ShowWindowAsync.

UpdateWindow ()
  Corresponds to CWindowBase::UpdateWindow.

ValidateRect (lpRect)
  Corresponds to CWindowBase::ValidateRect.

ValidateRgn (hRgn)
  Corresponds to CWindowBase::ValidateRgn.

WinHelp (args?)
  Corresponds to CWindowBase::WinHelp.

cConstructor (!args?=nil)
  Corresponds to CWindowBase::cConstructor.
destroyer ()
    Corresponds to CWindowBase::destructor.

onChange (sym, fn)
    Corresponds to CWindowBase::onChange.

This class also inherits the functions of
# Global Functions

## A
- `AbortDoc`
- `AbortPath`
- `ActivateKeyboardLayout`
- `AddFontResource`
- `AdjustWindowRect`
- `AdjustWindowRectEx`
- `AllowSetForegroundWindow`
- `AlphaBlend`
- `AngleArc`
- `AnimatePalette`
- `AnimateWindow`
- `AppendMenu`
- `Arc`
- `ArcTo`
- `ArrangeIconicWindows`
- `AttachThreadInput`

## B
- `BeginDeferWindowPos`
- `BeginPath`
- `BitBlt`
- `BringWindowToTop`

## C
- `CallMsgFilter`
- `CallNextHookEx`
- `CancelDC`
- `ChangeClipboardChain`
- `ChangeMenu`
- `CharLower`
- `CharLowerBuff`
- `CharNext`
- `CharNextEx`
- `CharPrev`
- `CharPrevEx`
- `CharToOem`
- `CharToOemBuff`
- `CharUpper`
- `CharUpperBuff`
- `CloseFigure`
- `CloseMetaFile`
- `CreateFont`
- `CreateFontDirectEx`
- `CreateFontIndirect`
- `CreateFontIndirectEx`
- `CreateHalftonePalette`
- `Create HatchBrush`
- `CreateIconIndirect`
- `CreateIC`
- `CreateMappedBitmap`
- `CreateMdiWindow`
- `CreateMenupanel`
- `CreateMenu`
- `CreatePatternBrush`
- `CreatePen`
- `CreatePenIndirect`
- `CreatePolyline`
- `CreatePolyPolygon`
- `CreatePopupMenu`
- `CreateRect`
- `CreateRectRgn`
- `CreateRectRgnIndirect`
- `CreateRoundRectRgn`
- `CreateSolidBrush`
- `CreateStatusWindow`
- `CreateToolBarEx`
- `CreateUpDownControl`
- `CreateEnhMetaFile`
- `CreateHalftonePalette`
- `CreateHatchBrush`
- `CreateIconIndirect`
- `CreateIC`
- `CreateMappedBitmap`
- `CreateMdiWindow`
- `CreateMenupanel`
- `CreateMenu`
- `CreatePatternBrush`
- `CreatePen`
- `CreatePenIndirect`
- `CreatePolyline`
- `CreatePolyPolygon`
- `CreatePopupMenu`
- `CreateRect`
- `CreateRectRgn`
- `CreateRectRgnIndirect`
- `CreateRoundRectRgn`
- `CreateSolidBrush`
- `CreateStatusWindow`
- `CreateToolBarEx`
- `CreateUpDownControl`
- `CreateEnhMetaFile`
Global Functions

D

DefDlgProc  DestroyAcceleratorTable  DrawAnimatedRects
DeferWindowPos  DestroyCaret  DrawCaption
DefFrameProc  DestroyCursor  DrawEdge
DefMDIChildProc  DestroyIcon  DrawEscape
DefSubclassProc  DestroyMenu  DrawFocusRect
DefWindowProc  DestroyWindow  DrawFrameControl
DeleteColorSpace  DeviceCapabilities  DrawIcon
DeleteDC  DispatchMessage  DrawIconEx
DeleteEnhMetaFile  DlgDirList  DrawInsert
DeleteMenu  DlgDirListComboBox  DrawMenuBar
DeleteMetaFile  DlgDirSelectComboBoxEx  DrawStatusText
DeleteObject  DlgDirSelectEx  DrawText
DeregisterShellHookWindow  DragDetect  DrawTextEx
DescribePixelFormat  DragObject

E

Ellipse  EndMenu  ExcludeClipRect
EmptyClipboard  EndPage  ExcludeUpdateRgn
EnableMenuIem  EndPaint  ExitWindowsEx
EnableScrollBar  EndPath  ExtCreateRegion
EnableWindow  EnumClipboardFormats  ExtEscape
EndDeferWindowPos  EnumWindows  ExtFloodFill
EndDialog  EqualRect  ExtSelectClipRgn
EndDoc  EqualRgn

F

FillPath  FlashWindow  FlatSB_SetScrollRange
FillRect  FlatSB_EnableScrollBar  FlatSB_ShowScrollBar
FillRgn  FlatSB_GetScrollInfo  FlattenPath
FindPixelInArea  FlatSB_GetScrollPos  FloodFill
FindWindow  FlatSB_SetScrollInfo  FrameRect
FindWindowEx  FlatSB_SetScrollPos  FrameRgn
FixBrushOrgEx  FlatSB_SetScrollProp

G

GdiFlush  GetEnhMetaFileHeader  GetPixel
GdiGetBatchLimit  GetEnhMetaFilePaletteEntries  GetPixelFormat
GdiSetBatchLimit  GetEnhMetaFilePixelFormat  GetPolyFillMode
GetActiveWindow  GetFileTitleA  GetProcessWindowStation
GetAltTabInfo  GetFocus  GetProp
GetAncestor  GetFontLanguageInfo  GetQueueStatus
<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetArcDirection</td>
<td>GetFontUnicodeRanges</td>
<td>GetRandomRgn</td>
</tr>
<tr>
<td>GetAspectRatioFilterEx</td>
<td>GetForegroundWindow</td>
<td>GetRasterizerCaps</td>
</tr>
<tr>
<td>GetAsyncKeyState</td>
<td>GetGraphicsMode</td>
<td>GetRegionData</td>
</tr>
<tr>
<td>GetBitmapDimensionEx</td>
<td>GetGUIResources</td>
<td>GetRgnBox</td>
</tr>
<tr>
<td>GetBkColor</td>
<td>GetGUIThreadInfo</td>
<td>GetROP2</td>
</tr>
<tr>
<td>GetBkMode</td>
<td>GetIconInfo</td>
<td>GetScrollBarInfo</td>
</tr>
<tr>
<td>GetBoundsRect</td>
<td>GetInputState</td>
<td>GetScrollInfo</td>
</tr>
<tr>
<td>GetBrushOrgEx</td>
<td>GetKBCodePage</td>
<td>GetScrollPos</td>
</tr>
<tr>
<td>GetCapture</td>
<td>GetKerningPairs</td>
<td>GetShellWindow</td>
</tr>
<tr>
<td>GetCaretBlinkTime</td>
<td>GetKeyboardLayout</td>
<td>GetStockObject</td>
</tr>
<tr>
<td>GetCaretPos</td>
<td>GetKeyboardLayoutName</td>
<td>GetStretchBltMode</td>
</tr>
<tr>
<td>GetCharABCWidths</td>
<td>GetKeyboardType</td>
<td>GetSubMenu</td>
</tr>
<tr>
<td>GetCharABCWidthsFloat</td>
<td>GetKeyNameText</td>
<td>GetSysColor</td>
</tr>
<tr>
<td>GetCharacterPlacement</td>
<td>GetKeyState</td>
<td>GetSysColorBrush</td>
</tr>
<tr>
<td>GetClassInfo</td>
<td>GetLastActivePopup</td>
<td>GetSystemMenu</td>
</tr>
<tr>
<td>GetClassInfoEx</td>
<td>GetLastInputInfo</td>
<td>GetSystemMetrics</td>
</tr>
<tr>
<td>GetClassLong</td>
<td>GetLayout</td>
<td>GetSystemPaletteEntries</td>
</tr>
<tr>
<td>GetClassName</td>
<td>GetListBoxInfo</td>
<td>GetSystemPaletteUse</td>
</tr>
<tr>
<td>GetClassWord</td>
<td>GetLogColorSpace</td>
<td>GetTextAlign</td>
</tr>
<tr>
<td>GetClientRect</td>
<td>GetMapMode</td>
<td>GetTextCharacterExtra</td>
</tr>
<tr>
<td>GetClipboardData</td>
<td>GetMenu</td>
<td>GetTextCharset</td>
</tr>
<tr>
<td>GetClipboardFormatName</td>
<td>GetMenuBarInfo</td>
<td>GetTextCharsetInfo</td>
</tr>
<tr>
<td>GetClipboardOwner</td>
<td>GetMenuCheckMarkDimensions</td>
<td>GetTextColor</td>
</tr>
<tr>
<td>GetClipboardSequenceNumber</td>
<td>GetMenuContextHelpId</td>
<td>GetTextExtentPoint</td>
</tr>
<tr>
<td>GetClipboardViewer</td>
<td>GetMenuDefaultItem</td>
<td>GetTextExtentPoint32</td>
</tr>
<tr>
<td>GetClipBox</td>
<td>GetMenuInfo</td>
<td>GetTextFace</td>
</tr>
<tr>
<td>GetClipCursor</td>
<td>GetMenuItemCount</td>
<td>GetTextMetrics</td>
</tr>
<tr>
<td>GetClipRgn</td>
<td>GetMenuItemID</td>
<td>GetThreadDesktop</td>
</tr>
<tr>
<td>GetColorAdjustment</td>
<td>GetMenuItemInfo</td>
<td>GetTitleBarInfo</td>
</tr>
<tr>
<td>GetColorSpace</td>
<td>GetMenuItemRect</td>
<td>GetTopWindow</td>
</tr>
<tr>
<td>GetComboBoxInfo</td>
<td>GetMenuState</td>
<td>GetUpdateRect</td>
</tr>
<tr>
<td>GetCurrentObject</td>
<td>GetMenuString</td>
<td>GetUpdateRgn</td>
</tr>
<tr>
<td>GetCurrentPositionEx</td>
<td>GetMessage</td>
<td>GetViewportExtEx</td>
</tr>
<tr>
<td>GetCursor</td>
<td>GetMessageExtraInfo</td>
<td>GetViewportOrgEx</td>
</tr>
<tr>
<td>GetCursorInfo</td>
<td>GetMessagePos</td>
<td>GetWindow</td>
</tr>
<tr>
<td>GetCursorPos</td>
<td>GetMessageTime</td>
<td>GetWindowContextHelpId</td>
</tr>
<tr>
<td>GetDC</td>
<td>GetMetaFile</td>
<td>GetWindowDC</td>
</tr>
<tr>
<td>GetDCBrushColor</td>
<td>GetMetaRgn</td>
<td>GetWindowExtEx</td>
</tr>
<tr>
<td>GetDCEx</td>
<td>GetMonitorInfo</td>
<td>GetWindowInfo</td>
</tr>
<tr>
<td>GetDCOrgEx</td>
<td>GetMouseMovePointsEx</td>
<td>GetWindowLong</td>
</tr>
<tr>
<td>GetDCPenColor</td>
<td>GetMUILanguage</td>
<td>GetWindowModuleFileName</td>
</tr>
<tr>
<td>GetDesktopWindow</td>
<td>GetNearestColor</td>
<td>GetWindowOrgEx</td>
</tr>
<tr>
<td>GetDeviceCaps</td>
<td>GetNearestPaletteIndex</td>
<td>GetWindowPlacement</td>
</tr>
<tr>
<td>GetDlgItemBaseUnits</td>
<td>GetNextDlgGroupItem</td>
<td>GetWindowRect</td>
</tr>
<tr>
<td>GetDIBColorTable</td>
<td>GetNextDlgTabItem</td>
<td>GetWindowRgn</td>
</tr>
<tr>
<td>GetDlgCtrlID</td>
<td>GetObjectType</td>
<td>GetWindowRgnBox</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>GetDlgItem</td>
<td>GetOpenClipboardWindow</td>
<td>GetWindowText</td>
</tr>
<tr>
<td>GetDlgItemText</td>
<td>GetOpenFileNameW</td>
<td>GetWindowTextLength</td>
</tr>
<tr>
<td>GetDoubleClickTime</td>
<td>GetOutlineTextMetrics</td>
<td>GetWindowWord</td>
</tr>
<tr>
<td>GetEnhMetaFile</td>
<td>GetPaletteEntries</td>
<td>GetWorldTransform</td>
</tr>
<tr>
<td>GetEnhMetaFileDescription</td>
<td>GetParent</td>
<td></td>
</tr>
</tbody>
</table>

H

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HideCaret</td>
</tr>
<tr>
<td>HiliteMenuItem</td>
</tr>
</tbody>
</table>

I

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageList_Add</td>
<td>ImageList_Merge</td>
<td>InvertRect</td>
</tr>
<tr>
<td>ImageList_AddMasked</td>
<td>ImageList_Remove</td>
<td>InvertRgn</td>
</tr>
<tr>
<td>ImageList_Begindrag</td>
<td>ImageList_Replace</td>
<td>IsCharAlpha</td>
</tr>
<tr>
<td>ImageList_Copy</td>
<td>ImageList_ReplaceIcon</td>
<td>IsCharAlphaNumeric</td>
</tr>
<tr>
<td>ImageList_Create</td>
<td>ImageList_SetBkColor</td>
<td>IsCharLower</td>
</tr>
<tr>
<td>ImageList_Destroy</td>
<td>ImageList_SetDragCursorImage</td>
<td></td>
</tr>
<tr>
<td>ImageList_DragEnter</td>
<td>ImageList_SetIconSize</td>
<td>IsChild</td>
</tr>
<tr>
<td>ImageList_DragLeave</td>
<td>ImageList_SetImageCount</td>
<td>IsClipboardFormatAvailable</td>
</tr>
<tr>
<td>ImageList_DragMove</td>
<td>ImageList_SetOverlayImage</td>
<td>IsDialogMessage</td>
</tr>
<tr>
<td>ImageList_DragShowNolock</td>
<td>InflateRect</td>
<td>IsDlgButtonChecked</td>
</tr>
<tr>
<td>ImageList_Draw</td>
<td>InitCommonControls</td>
<td>IsGUIThread</td>
</tr>
<tr>
<td>ImageList_DrawEx</td>
<td>InitCommonControlsEx</td>
<td>IsIconic</td>
</tr>
<tr>
<td>ImageList_DrawIndirect</td>
<td>InitializeFlatSB</td>
<td>IsMenu</td>
</tr>
<tr>
<td>ImageList_Duplicate</td>
<td>InitMUILanguage</td>
<td>IsRectEmpty</td>
</tr>
<tr>
<td>ImageList_EndDrag</td>
<td>InSendMessage</td>
<td>IsWindow</td>
</tr>
<tr>
<td>ImageList_GetBkColor</td>
<td>InsertMenu</td>
<td>IsWindowEnabled</td>
</tr>
<tr>
<td>ImageList_GetDragImage</td>
<td>InsertMenuItem</td>
<td>IsWindowUnicode</td>
</tr>
<tr>
<td>ImageList_GetIcon</td>
<td>IntersectMenuItem</td>
<td>IsWindowVisible</td>
</tr>
<tr>
<td>ImageList_GetImageCount</td>
<td>IntersectRect</td>
<td>IsWinEventHookInstalled</td>
</tr>
<tr>
<td>ImageList_GetImageInfo</td>
<td>InvalidateRect</td>
<td></td>
</tr>
<tr>
<td>ImageList_LoadImage</td>
<td>InvalidateRgn</td>
<td></td>
</tr>
</tbody>
</table>

K

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>keybd_event</td>
</tr>
<tr>
<td>KillTimer</td>
</tr>
</tbody>
</table>

L

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LineTo</td>
<td>LoadIcon</td>
<td>LoadString</td>
</tr>
<tr>
<td>LoadAccelerators</td>
<td>LoadImage</td>
<td>LockSetForegroundWindow</td>
</tr>
<tr>
<td>LoadBitmap</td>
<td>LoadImageID</td>
<td>LockWindowUpdate</td>
</tr>
<tr>
<td>LoadCursor</td>
<td>LoadKeyboardLayout</td>
<td>LockWorkStation</td>
</tr>
<tr>
<td>LoadCursorFromFile</td>
<td>LoadMenu</td>
<td></td>
</tr>
</tbody>
</table>
Global Functions

M

MakeDragList  MessageBeep  MonitorFromRect
MapDialogRect  MessageBox  MonitorFromWindow
MapVirtualKey  MessageBoxEx  mouse_event
MapVirtualKeyEx  MessageBoxIndirect  MoveToEx
MapWindowPoints  ModifyMenu  MoveWindow
MaskBlt  ModifyWorldTransform
MenuFromPoint  MonitorFromPoint

N

new__U_MENUorID  new_CMMenu  NotifyWinEvent
new__U_RECT  new_GColorDialog
new__U_STRINGorID  new_GFileDialog

O

OemKeyScan  OffsetRect  OpenClipboard
OemToChar  OffsetRgn  OpenIcon
OemToCharBuff  OffsetViewportOrgEx
OffsetClipRgn  OffsetWindowOrgEx

P

PaintDesktop  PlayMetaFile  PolyTextOut
PaintRgn  PlayMetaFileRecord  PostMessage
PatBlt  PlgBlt  PostQuitMessage
PathToRegion  PolyBezier  PostThreadMessage
PeekMessage  PolyBezierTo  PrintWindow
Pie  Polygon  PtInRect
PlayEnhMetaFile  Polyline  PtInRegion
PlayEnhMetaFileRecord  PolylineTo  PtVisible

R

RealChildWindowFromPoint  RegisterClipboardFormat  RemoveMenu
RealGetWindowClass  RegisterHotKey  RemoveProp
RealizePalette  RegisterRawInputDevices  ReplyMessage
Rectangle  RegisterShellHookWindow  ResetDC
RectInRegion  RegisterWindowMessage  ResizePalette
RectVisible  ReleaseCapture  RestoreDC
RedrawWindow  ReleaseDC  RoundRect
RegisterClass  RemoveFontMemResourceEx
RegisterClassEx  RemoveFontResource
### Global Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaveDC</td>
<td>SetDlgItemInt</td>
<td>SetScrollRange</td>
</tr>
<tr>
<td>ScaleViewportExtEx</td>
<td>SetDlgItemText</td>
<td>SetStretchBltMode</td>
</tr>
<tr>
<td>ScaleWindowExtEx</td>
<td>SetDoubleClickTime</td>
<td>SetSystemCursor</td>
</tr>
<tr>
<td>ScreenToClient</td>
<td>SetFocus</td>
<td>SetSystemPaletteUse</td>
</tr>
<tr>
<td>ScrollDC</td>
<td>SetForegroundWindow</td>
<td>SetTextAlign</td>
</tr>
<tr>
<td>ScrollWindow</td>
<td>SetGraphicsMode</td>
<td>SetTextColor</td>
</tr>
<tr>
<td>ScrollWindowEx</td>
<td>SetICMMode</td>
<td>SetTextCharacterExtra</td>
</tr>
<tr>
<td>SelectClipPath</td>
<td>SetICMProfile</td>
<td>SetTextJustification</td>
</tr>
<tr>
<td>SelectClipRgn</td>
<td>SetLastErrorEx</td>
<td>SetThreadDesktop</td>
</tr>
<tr>
<td>SelectObject</td>
<td>SetLayeredWindowAttributes</td>
<td>SetViewportExtEx</td>
</tr>
<tr>
<td>SelectPalette</td>
<td>SetLayout</td>
<td>SetViewportOrgEx</td>
</tr>
<tr>
<td>SendDlgItemMessage</td>
<td>SetMapMode</td>
<td>SetWindowContextHelpId</td>
</tr>
<tr>
<td>SendMessage</td>
<td>SetMapperFlags</td>
<td>SetWindowExtEx</td>
</tr>
<tr>
<td>SendNotifyMessage</td>
<td>SetMenu</td>
<td>SetWindowLong</td>
</tr>
<tr>
<td>SetActiveWindow</td>
<td>SetMenuContextHelpId</td>
<td>SetWindowOrgEx</td>
</tr>
<tr>
<td>SetArcDirection</td>
<td>SetMenuDefaultItem</td>
<td>SetWindowPlacement</td>
</tr>
<tr>
<td>SetBitmapDimensionEx</td>
<td>SetMenuInfo</td>
<td>SetWindowPos</td>
</tr>
<tr>
<td>SetBkColor</td>
<td>SetMenuItemInfo</td>
<td>SetWindowText</td>
</tr>
<tr>
<td>SetBkMode</td>
<td>SetMessageExtraInfo</td>
<td>SetWindowTextWord</td>
</tr>
<tr>
<td>SetBoundsRect</td>
<td>SetMessageQueue</td>
<td>SetWorldTransform</td>
</tr>
<tr>
<td>SetBrushOrgEx</td>
<td>SetMetaRgn</td>
<td>ShowCaret</td>
</tr>
<tr>
<td>SetCapture</td>
<td>SetPaletteEntries</td>
<td>ShowCursor</td>
</tr>
<tr>
<td>SetCaretBlinkTime</td>
<td>SetParent</td>
<td>ShowOwnedPopups</td>
</tr>
<tr>
<td>SetCaretPos</td>
<td>SetPixel</td>
<td>ShowScrollBar</td>
</tr>
<tr>
<td>SetClassLong</td>
<td>SetPixelFormat</td>
<td>ShowWindow</td>
</tr>
<tr>
<td>SetClassWord</td>
<td>SetPixelV</td>
<td>ShowWindowAsync</td>
</tr>
<tr>
<td>SetClipboardData</td>
<td>SetPolyFillMode</td>
<td>StartDoc</td>
</tr>
<tr>
<td>SetClipboardViewer</td>
<td>SetProcessDefaultLayout</td>
<td>StartPage</td>
</tr>
<tr>
<td>SetColorAdjustment</td>
<td>SetProcessWindowStation</td>
<td>StretchBlt</td>
</tr>
<tr>
<td>SetColorSpace</td>
<td>SetProp</td>
<td>StrokeAndFillPath</td>
</tr>
<tr>
<td>SetCursor</td>
<td>SetRect</td>
<td>StrokePath</td>
</tr>
<tr>
<td>SetCursorPos</td>
<td>SetRectEmpty</td>
<td>SubtractRect</td>
</tr>
<tr>
<td>SetDCBrushColor</td>
<td>SetRectRgn</td>
<td>SwapBuffers</td>
</tr>
<tr>
<td>SetDCPenColor</td>
<td>SetROP2</td>
<td>SwapMouseButton</td>
</tr>
<tr>
<td>SetDebugErrorLevel</td>
<td>SetScrollInfo</td>
<td>SwitchDesktop</td>
</tr>
<tr>
<td>SetDIBColorTable</td>
<td>SetScrollPos</td>
<td>SwitchToThisWindow</td>
</tr>
<tr>
<td>TextOut</td>
<td>TrackPopupMenuEx</td>
<td>TranslateMessage</td>
</tr>
<tr>
<td>TrackMouseEvent</td>
<td>TranslateAccelerator</td>
<td>TransparentBlt</td>
</tr>
<tr>
<td>TrackPopupMenu</td>
<td>TranslateMDISysAccel</td>
<td></td>
</tr>
</tbody>
</table>

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
<table>
<thead>
<tr>
<th>Function</th>
<th></th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnhookWindowsHookEx</td>
<td>UnrealizeObject</td>
<td>UpdateICMRegKey</td>
</tr>
<tr>
<td>UnhookWinEvent</td>
<td>UnregisterClass</td>
<td>UpdateWindow</td>
</tr>
<tr>
<td>UninitializeFlatSB</td>
<td>UnregisterDeviceNotification</td>
<td>UserHandleGrantAccess</td>
</tr>
<tr>
<td>UnionRect</td>
<td>UnregisterHotKey</td>
<td>UpdateColors</td>
</tr>
<tr>
<td>UnloadKeyboardLayout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidateRect</td>
<td>VkKeyScan</td>
<td>VkKeyScanEx</td>
</tr>
<tr>
<td>ValidateRgn</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Non-Widget Classes

A

_ABC ACCELERATOR ACCELERATORV2 ACCELERATORV3
_ABC_FLOAT ALT_TABINFO AXISINFO
_ACCEL ACCESS_TIMEOUT AXESLIST

B

_BITMAP BITMAPINFO BLEND_FUNCTION
_BITMAPCOREHEADER BITMAPINFOHEADER BSTINFO
_BITMAPCOREINFO BITMAPV4HEADER BUTTON_IMAGE_LIST
_BITMAPFILEHEADER BITMAPV5HEADER

C

_CBTT_CREATEWND CHOOSEFONTW COMBOBOXEXITEM
_CBTTACTIVATESTRUCT CIEXYZW COMBOBOXINFO
_CHARFORMAT CIEXYZTRIPLE COMPAREITEMSTRUCT
_CHARFORMAT2 CLIENTCREATESTRUCT COPYDATASTRUCT
_CHARRANGE CMENU CREATESTRUCT
_CHARSETINFO CMenuItemInfo CURSORINFO
_CHOOSECOLOR A COLORADJUSTMENT CURSORSHAPE
_CHOOSECOLOR W COLORADJUSTMENT CURSORSHAPE
_CHOOSERENDER A COLORMAP CW_PRET_STRUCTURE
_CHOOSERENDER W COLORMAP CW_PRET_STRUCTURE
_CHOOSERENDER A COLORSHEME CWPRETSTRUCT
_CHOOSERENDER W COLORSHEME CWPRETSTRUCT

D

_DBGHOOKINFO DIBSECTION DOCINFO
_DELETEITEMSTRUCT DISPLAY_DEVICE DRAGLISTINFO
_DISPLAYVECTOR DISPLAY_DEVICEW DRAWITEMSTRUCT
_DEVMODE DLGITEMTEMPLATE DRAWITEMSTRUCT
_DEVMODE W DLGITEMTEMPLATE DRAWITEMSTRUCT
_DEVNAMES DOCINFO DROPSTRUCT

E

_EDITBALLOONTIP EMRFRAMERGN EMRSET_BK_COLOR
<table>
<thead>
<tr>
<th>Class</th>
<th>Non-Widget Classes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDITSTREAM</td>
<td>EMRGDICOMMENT</td>
<td>EMRSETCOLORADJUSTMENT</td>
</tr>
<tr>
<td>EMR</td>
<td>EMRGLSBOUNDDEDREC</td>
<td>EMRSETCOLORSPACE</td>
</tr>
<tr>
<td>EMRABORTPATH</td>
<td>EMRGLSRECORD</td>
<td>EMRSETDIBITSTODEVICE</td>
</tr>
<tr>
<td>EMRALPHABLEND</td>
<td>EMRGRADIENTFILL</td>
<td>EMRSETICMPROFILE</td>
</tr>
<tr>
<td>EMRANGLEARC</td>
<td>EMRINVERTRGN</td>
<td>EMRSETMAPPERFLAGS</td>
</tr>
<tr>
<td>EMRARC</td>
<td>EMRLINETO</td>
<td>EMRSETMITERLIMIT</td>
</tr>
<tr>
<td>EMRBITBLT</td>
<td>EMRMASKBLT</td>
<td>EMRSETPALETTEENTRIES</td>
</tr>
<tr>
<td>EMRCOLORCORRECTPALETTE</td>
<td>EMRMODIFYWORLDTRANSFORM</td>
<td>EMRSTPIXELV</td>
</tr>
<tr>
<td>EMRCREATEMONOBRUSH</td>
<td>EMRPOLYDRAW16</td>
<td>EMRSTRETCHDIBITS</td>
</tr>
<tr>
<td>EMRCREATEPALETTE</td>
<td>EMRPOLYLNE</td>
<td>EMRTEXT</td>
</tr>
<tr>
<td>EMRCREATEPEN</td>
<td>EMRPOLYLNE16</td>
<td>EMRTRANSARENTBLT</td>
</tr>
<tr>
<td>EMRELLIPSE</td>
<td>EMRPOLYPOLYLINE</td>
<td>ENHMETAHEADER</td>
</tr>
<tr>
<td>EMREOF</td>
<td>EMRPOLYPOLYLINE16</td>
<td>ENHMETARECORD</td>
</tr>
<tr>
<td>EMREXCLUDECLIPRECT</td>
<td>EMRPOLYTEXOUTA</td>
<td>ENUMLOGFONT</td>
</tr>
<tr>
<td>EMREXTCREATEPEN</td>
<td>EMRRSIZEPALETTE</td>
<td>ENUMLOGFONTEX</td>
</tr>
<tr>
<td>EMREXTESCAPE</td>
<td>EMRRESTOREDC</td>
<td>ENUMLOGFONTEXDV</td>
</tr>
<tr>
<td>EMREXTFLOODFILL</td>
<td>EMRROUNDRECT</td>
<td>ENUMTEXMETRIC</td>
</tr>
<tr>
<td>EMREXTSELECTCLIPRGN</td>
<td>EMRSCALEVIEWPORTTEXTEX</td>
<td>EVENTMSG</td>
</tr>
<tr>
<td>EMREXTEXTXOUTA</td>
<td>EMRSELECTCLIPPATH</td>
<td>EXTLOGFONT</td>
</tr>
<tr>
<td>EMRFILLPATH</td>
<td>EMRSELECTOBJECT</td>
<td>EXTLOGPEN</td>
</tr>
<tr>
<td>EMRFILLRGN</td>
<td>EMRSELECTPALETTE</td>
<td></td>
</tr>
<tr>
<td>EMRFORMAT</td>
<td>EMRSETARCDIRECTION</td>
<td></td>
</tr>
</tbody>
</table>

**F**

<table>
<thead>
<tr>
<th>Class</th>
<th>Class</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILETIME</td>
<td>FINDTEXTEX</td>
<td>FONTSIGNATURE</td>
</tr>
<tr>
<td>FILTERKEYS</td>
<td>FINDTEXTEX</td>
<td>FORMATRANGE</td>
</tr>
<tr>
<td>FINDREPLACEA</td>
<td>FIXED</td>
<td></td>
</tr>
<tr>
<td>FINDREPLACEW</td>
<td>FLASHWININFO</td>
<td></td>
</tr>
</tbody>
</table>

**G**

<table>
<thead>
<tr>
<th>Class</th>
<th>Class</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCP_RESULTS</td>
<td>GLYPHMETRICS</td>
<td>GRADIENT_RECT</td>
</tr>
<tr>
<td>GETTEXTEX</td>
<td>GLYPHMETRICSFLOAT</td>
<td>GRADIENT_TRIANGLE</td>
</tr>
<tr>
<td>GETTEXTLENGTHEX</td>
<td>GLYPHSET</td>
<td>GUITHREADINFO</td>
</tr>
</tbody>
</table>

**H**

<table>
<thead>
<tr>
<th>Class</th>
<th>Class</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>HANDLETABLE</td>
<td>HD_TEXTFILTERW</td>
<td>HDLAYOUT</td>
</tr>
<tr>
<td>HARDWAREHOOKSTRUCT</td>
<td>HDHITTESTINFO</td>
<td>HELPINFO</td>
</tr>
<tr>
<td>HARDWAREINPUT</td>
<td>HDITEM</td>
<td>HELPWININFO</td>
</tr>
<tr>
<td>HD_TEXTFILTERA</td>
<td>HDITEMW</td>
<td>HIGHCONTRAST</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
<td>K</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>ICONINFO</td>
<td>IMAGELISTDRAWPARAMS</td>
<td>INPUT</td>
</tr>
<tr>
<td>LASTINPUTINFO</td>
<td>LOGFONT</td>
<td>LVHITTESTINFO</td>
</tr>
<tr>
<td>LITEM</td>
<td>LVBIKIMAGE</td>
<td>LVIEWITEM</td>
</tr>
<tr>
<td>LOGCOLORSPACE</td>
<td>LVIEWGROUP</td>
<td>LVTILEVIEWINFO</td>
</tr>
<tr>
<td>MAT2</td>
<td>MENUITEMTEMPLATE</td>
<td>MOUSEHOOKSTRUCT</td>
</tr>
<tr>
<td>MDINEXTMENU</td>
<td>METAHEADER</td>
<td>MOUSEKEYS</td>
</tr>
<tr>
<td>MENUOBJECTINFO</td>
<td>MINMAXINFO</td>
<td>MSGBOXPARAMS</td>
</tr>
<tr>
<td>NCCALCSIZE_PARAMS</td>
<td>NITEMACTIVATE</td>
<td>NMRBARCHILDSIZE</td>
</tr>
<tr>
<td>NMBCHOTITEM</td>
<td>NMLVCACHEHINT</td>
<td>NMTBDISPINFO</td>
</tr>
<tr>
<td>NMCCHAR</td>
<td>NMLVFINDITEM</td>
<td>NMTBRESTORE</td>
</tr>
<tr>
<td>NMDATETIMECHANGE</td>
<td>NMLVODSTATECHANGE</td>
<td>NMTICKBAR</td>
</tr>
<tr>
<td>Non-Widget Classes</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>NMDATETIMESTRING</td>
<td>NMOBJECTNOTIFY</td>
<td>NMTTDISPINFO</td>
</tr>
<tr>
<td>NMDATETIMEWMKEYDOWN</td>
<td>NMPGCALCSIZE</td>
<td>NMTVDISPINFO</td>
</tr>
<tr>
<td>NMDAYSTATE</td>
<td>NMPHOTITEM</td>
<td>NMTVDISPINFOW</td>
</tr>
<tr>
<td>NMHDDISPINFO</td>
<td>NMPGSCROLL</td>
<td>NMTVGETINFOTIP</td>
</tr>
<tr>
<td>NMHDFILTERBTNCLICK</td>
<td>NMRBAUTOFSIZE</td>
<td>NMTVKEYDOWN</td>
</tr>
<tr>
<td>NMHDR</td>
<td>NMREBAR</td>
<td>NMUPDOWN</td>
</tr>
<tr>
<td>NMHEADER</td>
<td>NMREBARAUTOBREAK</td>
<td></td>
</tr>
<tr>
<td>NMIPADDRESS</td>
<td>NMREBARCHEVERN</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>OFNOTIFYA</td>
<td>OPENFILENAME</td>
</tr>
<tr>
<td>O</td>
<td>OFNOTIFYEXA</td>
<td>OPENFILENAME_NT4A</td>
</tr>
<tr>
<td>O</td>
<td>OFNOTIFYEXW</td>
<td>OPENFILENAME_NT4W</td>
</tr>
<tr>
<td>O</td>
<td>OFNOTIFYW</td>
<td>OPENFILENAMEA</td>
</tr>
<tr>
<td>P</td>
<td>PAGESETUPDLGA</td>
<td>PBRANGE</td>
</tr>
<tr>
<td>P</td>
<td>PAINTSTRUCT</td>
<td>PIXELFORMATDESCRIPTOR</td>
</tr>
<tr>
<td>P</td>
<td>PALETTEENTRY</td>
<td>POINT</td>
</tr>
<tr>
<td>P</td>
<td>PANOS</td>
<td>POINTFLOAT</td>
</tr>
<tr>
<td>P</td>
<td>PARAFORMAT</td>
<td>POINTFX</td>
</tr>
<tr>
<td>P</td>
<td>PARAFORMAT2</td>
<td>POINTL</td>
</tr>
<tr>
<td>R</td>
<td>RASTERIZER_STATUS</td>
<td>RAWMOUSE</td>
</tr>
<tr>
<td>R</td>
<td>RAWHID</td>
<td>RBHITTESTINFO</td>
</tr>
<tr>
<td>R</td>
<td>RAWINPUT</td>
<td>REBARBANDINFO</td>
</tr>
<tr>
<td>R</td>
<td>RAWINPUT_data</td>
<td>REBARINFO</td>
</tr>
<tr>
<td>R</td>
<td>RAWINPUTDEVICE</td>
<td>RECT</td>
</tr>
<tr>
<td>R</td>
<td>RAWINPUTDEVICELIST</td>
<td>RECTL</td>
</tr>
<tr>
<td>R</td>
<td>RAWINPUTHEADER</td>
<td>RGBQUAD</td>
</tr>
<tr>
<td>R</td>
<td>RAWKEYBOARD</td>
<td>RGBTRIPLE</td>
</tr>
<tr>
<td>S</td>
<td>SCROLLBARINFO</td>
<td>SHELLHOOKINFO</td>
</tr>
<tr>
<td>S</td>
<td>SCROLLINFO</td>
<td>SIZE</td>
</tr>
<tr>
<td>S</td>
<td>SERIALKEYS</td>
<td>SOUNDSENTRY</td>
</tr>
<tr>
<td>T</td>
<td>TBADDBITMAP</td>
<td>TEXTMETRIC</td>
</tr>
<tr>
<td>T</td>
<td>TTPOLYCURVE</td>
<td></td>
</tr>
</tbody>
</table>

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
<table>
<thead>
<tr>
<th>TBBUTTON</th>
<th>TEXITRANGE</th>
<th>TTPOLYGONHEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBBUTTONINFO</td>
<td>TITLEBARINFO</td>
<td>TTTOOLINFO</td>
</tr>
<tr>
<td>TBINSERTMARK</td>
<td>TOGGLEKEYS</td>
<td>TTTOOLINFOW</td>
</tr>
<tr>
<td>TBMETRICS</td>
<td>TPMPARAMS</td>
<td>TVHITTESTINFO</td>
</tr>
<tr>
<td>TBREPLACEBITMAP</td>
<td>TRACKMOUSEEVENT</td>
<td>TVINSERTSTRUCT</td>
</tr>
<tr>
<td>TBSAVEPARAMS</td>
<td>TRIVERTEX</td>
<td>TVITEM</td>
</tr>
<tr>
<td>TCHITTESTINFO</td>
<td>TTGETTITLE</td>
<td>TVITEMEX</td>
</tr>
<tr>
<td>TCITEM</td>
<td>TTHITTESTINFO</td>
<td>TVSORTCB</td>
</tr>
<tr>
<td>TCITEMHEADER</td>
<td>TTHITTESTINFO</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U</th>
<th>USEROBJECTFLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDACCEL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>WNDCLASSEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>WC RANGE</td>
<td>WINDOWPLACEMENT</td>
</tr>
<tr>
<td>WGLSWAP</td>
<td>WINDOWPOS</td>
</tr>
<tr>
<td>WINDOWINFO</td>
<td>WNDCLASS</td>
</tr>
<tr>
<td></td>
<td>WTSSESSION_NOTIFICATION</td>
</tr>
</tbody>
</table>
Constants

(DialogSplitHelper::ATL_DISPID_DATAFIELD
(DialogSplitHelper::ATL_WM_OCC_LOAD_FROMSTREAM
(DialogSplitHelper::ATL_DISPID_DATASOURCE
(DialogSplitHelper::ATL_WM_OCC_LOAD_FROMSTREAM_EX
(DialogSplitHelper::ATL_WM_OCC_INITNEW
(DialogSplitHelper::ATL_WM_OCC_LOAD_FROMSTORAGE
(DialogSplitHelper::ATL_WM_OCC_LOAD_FROMSTORAGE_EX

A

ABORTDOC
ABSOLUTE
AC_SRC_ALPHA
AC_SRC_OVER
ACCESS_FILTERKEYS
ACCESS_MOUSEKEYS
ACCESS_STICKYKEYS
ACM_OPEN
ACM_OPENA
ACM_OPENW
ACM_PLAY
ACM_STOP
ACN_START
ACN_STOP
ACS_AUTOPLAY
ACS_CENTER
ACS_TIMER
ACS_TRANSPARENT
AD_CLOCKWISE
AD_COUNTERCLOCKWISE
ALERT_SYSTEM_CRITICAL
ALERT_SYSTEM_ERROR
ALERT_SYSTEM_INFORMATIONAL
ALERT_SYSTEM_QUERY
ALERT_SYSTEM_WARNING
ALTERNATE
ANSI_CHARSET
ANSI_FIXED_FONT
ANSI_VAR_FONT
ANTIALIASED_QUALITY
APPCOMMAND_MEDIA_NEXTTRACK
APPCOMMAND_MEDIA_PAUSE
APPCOMMAND_MEDIA_PLAY
APPCOMMAND_MEDIA_PLAY_PAUSE
APPCOMMAND_MEDIA_PREVIOUSTRACK
APPCOMMAND_MEDIA_RECORD
APPCOMMAND_MEDIA_REWIND
APPCOMMAND_MEDIA_STOP
APPCOMMAND_MIC_ON_OFF_TOGGLE
APPCOMMAND_MICROPHONE_VOLUME_DOWN
APPCOMMAND_MICROPHONE_VOLUME_MUTE
APPCOMMAND_MICROPHONE_VOLUME_UP
APPCOMMAND_NEW
APPCOMMAND_OPEN
APPCOMMAND_PASTE
APPCOMMAND_PRINT
APPCOMMAND_REDO
APPCOMMAND_REPLY_TO_MAIL
APPCOMMAND_SEND_MAIL
APPCOMMAND_SAVE
APPCOMMAND_SPELL_CHECK
APPCOMMAND_TREBLE_DOWN
APPCOMMAND_TREBLE_UP
APPCOMMAND_UNDO
APPCOMMAND_VOLUME_DOWN
APPCOMMAND_VOLUME_MUTE
APPCOMMAND_VOLUME_UP
ARABIC_CHARSET
ARW_BOTTOMLEFT
ARW_BOTTOMRIGHT

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
APPCOMMAND_BASS_BOOST ARW_DOWN
APPCOMMAND_BASS_DOWN ARW_HIDE
APPCOMMAND_BASS_UP ARW_LEFT
APPCOMMAND_BROWSER_BACKWARD ARW_RIGHT
APPCOMMAND_BROWSER_FAVORITES ARW_STARTMASK
APPCOMMAND_BROWSER_FORWARD ARW_STARTRIGHT
APPCOMMAND_BROWSER_HOME ARW_STARTTOP
APPCOMMAND_BROWSER_REFRESH ARW_TOOLEFT
APPCOMMAND_BROWSER_SEARCH ARW_TOPRIGHT
APPCOMMAND_BROWSER_STOP ARW_UP
APPCOMMAND_CLOSE ASPECT_FILTERING
APPCOMMAND_COPY ASPECTX
APPCOMMAND_CORRECTION_LIST ASPECTXY
APPCOMMAND_CUT ASPECTY
APPCOMMAND_DICTATE_OR_COMMAND_CONTROL_TOGGLE ATF_ONOFFFEEDBACK
APPCOMMAND_FIND ATF_TIMEOUTON
APPCOMMAND_FORWARD_MAIL AW_ACTIVATE
APPCOMMAND_HELP AW_BLEND
APPCOMMAND_LAUNCH_APP1 AW_CENTER
APPCOMMAND_LAUNCH_APP2 AW_HIDE
APPCOMMAND_LAUNCH_MAIL AW_HOR_NEGATIVE
APPCOMMAND_LAUNCH_MEDIA_SELECT AW_HOR_POSITIVE
APPCOMMAND_MEDIA_CHANNEL_DOWN AW_SLIDE
APPCOMMAND_MEDIA_CHANNEL_UP AW_VER_NEGATIVE
APPCOMMAND_MEDIA_FAST_FORWARD AW_VER_POSITIVE

B

BALTIC_CHARSET BS_CHECKBOX
BANDINFO BS_DEFPUSHBUTTON
BCM_FIRST BS_DIBPATTERN
BCM_GETIDEALSIZE BS_DIBPATTERN8X8
BCM_GETIMAGELIST BS_DIBPATTERNPT
BCM_GETTEXTMARGIN BS_FLAT
BCM_SETIMAGELIST BS_GROUPBOX
BCM_SETTEXTMARGIN BS_HATCHED
BCN_HOTITEMCHANGE BS_HOLLOW
BDR_INNER BS_ICON
BDR_OUTER BS_INDEXED
BDR_RAISED BS_LEFT
BDR_RAISEDINNER BS_LEFTTEXT
BDR_RAISEDOUTER BS_MONOPATTERN
BDR_SUNKEN BS_MULTILINE
BDR_SUNKENINNER BS_NOTIFY
BDR_SUNKENOUTER BS_NULL
BEGIN_PATH BS_OWNERDRAW
BF_ADJUST BS_PATTERN
BF_BOTTOM BS_PATTERN8X8
BF_BOTTOMLEFT BS_PUSHBOX
BF_BOTTOMRIGHT BS_PUSHBUTTON
BF_DIAGONAL BS_PUSHLIKE
BF_DIAGONAL_ENDBOTTOMLEFT BS_RADIOBUTTON
BF_DIAGONAL_ENDBOTTOMRIGHT BS_RIGHT
BF_DIAGONAL_ENDTOPLEFT BS_RIGHTBUTTON
BF_DIAGONAL_ENDTOPRIGHT BS_SOLID
BF_FLAT BS_TEXT
BF_LEFT BS_TOP
BF_MIDDLE BS_TYPEMASK
BF_MONO BS_USERBUTTON
BF_RECT BS_VCENTER
BF_RIGHT BSF_ALLOWSFW
BF_SOFT BSF_FLUSHDISK
BF_TOP BSF_FORCEIFHUNG
BF_TOPLEFT BSF_IGNORECURRENTTASK
BF_TOPRIGHT BSF_LUID
BITSPixel BSF_NOHANG
BKMODE_LAST BSF_NOTIMEOUTIFNOTHUNG
BLACK_BRUSH BSF_POSTMESSAGE
BLACK_PEN BSF_QUERY
BLACKONWHITE BSF_RETURNHDDESK
BLTALIGNMENT BSF_SENDNOTIFYMESSAGE
BM_CLICK BSM_ALLCOMPONENTS
BM_GETCHECK BSM_ALLDESKTOPS
BM_GETIMAGE BSM_APPLICATIONS
BM_GETSTATE BSM_INSTALLABLEDRIVERS
BM_SETCHECK BSM_NETDRIVER
BM_SETIMAGE BSM_VXDS
BM_SETSTATE BST_CHECKED
BM_SETSTYLE BST_FOCUS
BN_CLICKED BST_HOT
BN_DOUBLECLICKED BST_INDETERMINATE
BN_DISABLE BST_PUSHED
BN_DOUBLECLICKED BST_UNCHECKED
BN_HILITE BTNS_AUTOSIZE
BN_KILLFOCUS BTNS_BUTTON
BN_PAINT BTNS_CHECK
BN_PRESSED BTNS_CHECKGROUP
BN_SETFOCUS BTNS_DROPDOWN
BN_UNHILITE BTNS_GROUP
BN_UNPUSHED BTNS_NOPREFIX
BOLD_FONTTYPE BTNS_SEP
BROADCAST_QUERY_DENY
BS_3STATE
BS_AUTOSTATE
BS_AUTOCHECKBOX
BS_AUTORADIOBUTTON
BS_BITMAP
BS_BOTTOM
BS_CENTER

BUFFER_QUERY_DENY
BTNS_SHOWTEXT
BTNS_WHOLEDROPDOWN
BUTTON_IMAGELIST_ALIGN_BOTTOM
BUTTON_IMAGELIST_ALIGN_CENTER
BUTTON_IMAGELIST_ALIGN_LEFT
BUTTON_IMAGELIST_ALIGN_RIGHT
BUTTON_IMAGELIST_ALIGN_TOP

CA_LOG_FILTER
CA_NEGATIVE
CALERT_SYSTEM
CB_ADDSTRING
CB_DELETESTRING
CB_DIR
CB_ERR
CB_ERRSPACE
CB_FINDSTRING
CB_FINDSTRINGEXACT
CB_GETCOMBOBOXINFO
CB_GETCOUNT
CB_GETCURSEL
CB_GETDROPPEDCONTROLRECT
CB_GETDROPPEDSTATE
CB_GETDROPPEDWIDTH
CB_GETEDITSEL
CB_GETEXTENDEDUI
CB_GETHORIZONTALEXTENT
CB_GETITEMDATA
CB_GETITEMHEIGHT
CB_GETLBTEXT
CB_GETLBTEXTLEN
CB_GETLOCALE
CB_GETMINVISIBLE
CB_GETTOPINDEX
CB_INITSTORAGE
CB_INSERTSTRING
CB_LIMITTEXT
CB_MSGMAX
CB_MULTIPLEADDSTRING
CB_OKAY
CB_RESETCONTENT
CB_SELECTSTRING
CB_SETCURSEL

CDIS_FOCUS
CDIS_GRAYED
CDIS_HOT
CDIS_INDETERMINATE
CDIS_MARKED
CDIS_SELECTED
CDIS_SHOWKEYBOARDCUES
CDM_FIRST
CDM_GETFILEPATH
CDM_GETFOLDERIDLIST
CDM_GETFOLDERPATH
CDM_GETSPEC
CDM_HIDECONTROL
CDM_LAST
CDM_SETCONTROLTEXT
CDM_SETDEFAULT
CDN_FILEOK
CDN_FOLDERCHANGE
CDN_HELP
CDN.IncludeItem
CDN_INI_DONE
CDN_SELCHANGE
CDN_SHAREVIOLATION
CDN>TypeChange
CDRF_DODEFAULT
CDRF_NEWFONT
CDRF_NOTIFYITEMDRAW
CDRF_NOTIFYPOSTERASE
CDRF_NOTIFYPOSTPAINT
CDRF_NOTIFYSUBITEMDRAW
CDRF_SKIPDEFAULT
CDS_FULLSCREEN
CDS_GLOBAL
CDS_NORESET
CDS_RESET
<p>| CB_SETDROPPEDWIDTH                  | CDS_SET_PRIMARY                        |
| CB_SETEDITSEL                       | CDS_TEST                                |
| CB_SETTEXTENDEDUI                   | CDS_UPDATEREGISTRY                     |
| CB_SETHORIZONTALEXTENT             | CDS_VIDEOPARAMETERS                    |
| CB_SETITEMDATA                      | CF_BITMAP                              |
| CB_SETITEMHEIGHT                   | CF_BOTH                                 |
| CB_SETLOCALE                       | CF_DIB                                 |
| CB_SETMINVISIBLE                   | CF_DIBV5                                |
| CB_SETTOPINDEX                     | CF_DIF                                 |
| CB_SHOWDROPPDOWN                   | CF_DSPBITMAP                            |
| CBEIF_DL_SETITEM                   | CF_DSPENHMETAFILE                      |
| CBEIF_IMAGE                         | CF_DSPMETAFILEPICT                     |
| CBEIF_INDENT                        | CF_DSPTEXT                              |
| CBEIF_LPARAM                        | CF_ENHMETAFILE                          |
| CBEIF_OVERLAY                       | CF_GDIOBJFIRST                         |
| CBEIF_SELECTEDIMAGE                | CF_GDIOBJLAST                           |
| CBEIF_TEXT                          | CF_HDROP                               |
| CBEM_DELETEITEM                    | CF_LOCALE                              |
| CBEM_GETCOMBOCONTROL              | CF_MAX                                 |
| CBEM_GETEDITCONTROL                | CF_METAFILEPICT                         |
| CBEM_GETEXSTYLE                    | CF_OEMTEXT                              |
| CBEM_GETEXTENDEDSTYLE              | CF_OWNERDISPLAY                         |
| CBEM_GETIMAGELIST                  | CF_PALETTE                              |
| CBEM_GETITEM                        | CF_PENDATA                             |
| CBEM_GETITEMA                      | CF_PRINTERFONTS                        |
| CBEM_GETITEMW                      | CF_PRIVATEFIRST                        |
| CBEM_GETUNICODEFORMAT              | CF_PRIVATELAST                         |
| CBEM_HASEDITCHANGED                | CF_RIFF                                |
| CBEM_INSERTITEM                    | CF_SCREENFONTS                         |
| CBEM_INSERTITEMA                   | CF_SYLK                                 |
| CBEM_INSERTITEMW                   | CF_TEXT                                 |
| CBEM_SETEXTYLE                     | CF_TIFF                                |
| CBEM_SETEXTYLE                     | CF_UNICODETEXT                         |
| CBEM_SETIMAGELIST                  | CF_WAVE                                |
| CBEM_SETITEM                       | CHECKJPEGFORMAT                        |
| CBEM_SETITEMA                      | CHECKPNGFORMAT                         |
| CBEM_SETITEMW                      | CHILDID_SELF                            |
| CBEM_SETUNICODEFORMAT              | CHINESEBIG5_CHARSET                    |
| CBEM_SetWindowTheme                | CLEARTYPE_NATURAL_QUALITY              |
| CBEMAXSTRLEN                       | CLEARTYPE_QUALITY                      |
| CBENN_BEGINEDIT                    | CLIP_CHARACTER_PRECIS                  |
| CBENN_DELETEITEM                   | CLIP_DEFAULT_PRECIS                    |
| CBENN_DRAGBEGIN                   | CLIP_EMBEDDED                          |
| CBENN_DRAGBEGINA                  | CLIP_LH_ANGLES                         |
| CBENN_DRAGBEGINW                  | CLIP_MASK                              |
| CBENN_ENDEDIT                      | CLIP_STROKE_PRECIS                     |</p>
<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBEN_ENDEDITA</td>
<td>CLIP_TO_PATH</td>
</tr>
<tr>
<td>CBEN_ENDEDITW</td>
<td>CLIP_TT_ALWAYS</td>
</tr>
<tr>
<td>CBEN_GETDISPINFOA</td>
<td>CLIPCAPS</td>
</tr>
<tr>
<td>CBEN_GETDISPINFOW</td>
<td>CLOSECHANNEL</td>
</tr>
<tr>
<td>CBEN_INSERTITEM</td>
<td>CLR_DEFAULT</td>
</tr>
<tr>
<td>CBENF_DROPDOWN</td>
<td>CLR_HILIGHT</td>
</tr>
<tr>
<td>CBENF_ESCAPE</td>
<td>CLR_INVALID</td>
</tr>
<tr>
<td>CBENF_KILLFOCUS</td>
<td>CLR_NONE</td>
</tr>
<tr>
<td>CBENF_RETURN</td>
<td>CM_CMYK_COLOR</td>
</tr>
<tr>
<td>CBES_EX_CASESENSITIVE</td>
<td>CM_DEVICE_ICM</td>
</tr>
<tr>
<td>CBES_EX_NOEDITIMAGE</td>
<td>CM_GAMMA_RAMP</td>
</tr>
<tr>
<td>CBES_EX_NOEDITIMAGEINDENT</td>
<td>CM_IN_GAMUT</td>
</tr>
<tr>
<td>CBES_EX_NOSIZELIMIT</td>
<td>CM_NONE</td>
</tr>
<tr>
<td>CBES_EX_PATHWORDBREAKPROC</td>
<td>CM_OUT_OF_GAMUT</td>
</tr>
<tr>
<td>CBM_FIRST</td>
<td>CMB_MASKED</td>
</tr>
<tr>
<td>CBN_CLOSEUP</td>
<td>COLOR_3DDKSHADOW</td>
</tr>
<tr>
<td>CBN_DBLCLK</td>
<td>COLOR_3DFACE</td>
</tr>
<tr>
<td>CBN_DROPDOWN</td>
<td>COLOR_3DHIGHLIGHT</td>
</tr>
<tr>
<td>CBN_EDITCHANGE</td>
<td>COLOR_3DHIghLIGHT</td>
</tr>
<tr>
<td>CBN_EDITUPDATE</td>
<td>COLOR_3DLIGHT</td>
</tr>
<tr>
<td>CBN_ERRSPACE</td>
<td>COLOR_3DSHADOW</td>
</tr>
<tr>
<td>CBN_KILLFOCUS</td>
<td>COLOR_ACTIVEBORDER</td>
</tr>
<tr>
<td>CBN_SELCHANGE</td>
<td>COLOR_ACTIVECAPTION</td>
</tr>
<tr>
<td>CBN_SELENDCANCEL</td>
<td>COLOR_APPWORKSPACE</td>
</tr>
<tr>
<td>CBN_SELENDOK</td>
<td>COLOR_BACKGROUND</td>
</tr>
<tr>
<td>CBN_SETFOCUS</td>
<td>COLOR_BTNFACE</td>
</tr>
<tr>
<td>CBS_AUTOHSCROLL</td>
<td>COLOR_BTNHIGHLIGHT</td>
</tr>
<tr>
<td>CBS_DISABLENOSCROLL</td>
<td>COLOR_BTNHIGHLIGHT</td>
</tr>
<tr>
<td>CBS_DROPDOWN</td>
<td>COLOR_BTNSHADOW</td>
</tr>
<tr>
<td>CBS_DROPDOWNLIST</td>
<td>COLOR_BTNTEXT</td>
</tr>
<tr>
<td>CBS_HASSTRINGS</td>
<td>COLOR_CAPTIONTEXT</td>
</tr>
<tr>
<td>CBS_LOWERCASE</td>
<td>COLOR_DESKTOP</td>
</tr>
<tr>
<td>CBS_NOINTEGRALHEIGHT</td>
<td>COLOR_GRADIENTACTIVECAPTION</td>
</tr>
<tr>
<td>CBS_OEMCONVERT</td>
<td>COLOR_GRADIENTINACTIVECAPTION</td>
</tr>
<tr>
<td>CBS_OWNERDRAWFIXED</td>
<td>COLOR_GRAYTEXT</td>
</tr>
<tr>
<td>CBS_OWNERDRAW_VARIABLE</td>
<td>COLOR_HIGHLIGHT</td>
</tr>
<tr>
<td>CBS_SIMPLE</td>
<td>COLOR_HIGHLIGHTTEXT</td>
</tr>
<tr>
<td>CBS_SORT</td>
<td>COLOR_HOTLIGHT</td>
</tr>
<tr>
<td>CBS_UPPERCASE</td>
<td>COLOR_INACTIVEBORDER</td>
</tr>
<tr>
<td>CC_ANYCOLOR</td>
<td>COLOR_INACTIVECAPTION</td>
</tr>
<tr>
<td>CC_CHORD</td>
<td>COLOR_INACTIVECAPTIONTEXT</td>
</tr>
<tr>
<td>CC_CIRCLES</td>
<td>COLOR_INFOBK</td>
</tr>
<tr>
<td>CC_ELLIPSES</td>
<td>COLOR_INFOTEXT</td>
</tr>
<tr>
<td>CC_ENABLEHOOK</td>
<td>COLOR_MENU</td>
</tr>
<tr>
<td>CC_ENABLETEMPLATE</td>
<td>COLOR_MENUBAR</td>
</tr>
<tr>
<td>CC_ENABLETEMPLATEHANDLE</td>
<td>COLOR_MENUGHILIGHT</td>
</tr>
</tbody>
</table>
CC_FULLOPEN
CC_INTERIORS
CC_NONE
CC_PIE
CC_PREVENTFULLOPEN
CC_RGBINIT
CC_ROUNDDIRECT
CC_SHOWHELP
CC_SOLIDCOLOR
CC_STYLED
CC_WIDE
CC_WIDESTYLED
CCHILDREN_SCROLLBAR
CCHILDREN_TITLEBAR
CCM_DPISCALE
CCM_FIRST
CCM_GETCOLORSCHEME
CCM_GETDROPTARGET
CCM_GETUNICODEFORMAT
CCM_GETVERSION
CCM_LAST
CCM_SETBKCOLOR
CCM_SETCOLORSCHEME
CCM_SETNOTIFYWINDOW
CCM_SETUNICODEFORMAT
CCM_SETVERSION
CCM_SETWINDOWTHEME
CD_LBSELADD
CD_LBSELCHANGE
CD_LBSELNOITEMS
CD_LBSELSEL
CDdds_ITEM
CDdds_ITEMPOSTERASE
CDdds_ITEMPOSTPAINT
CDdds_ITEMPREERASE
CDdds_ITEMPREPAINT
CDdds_POSTERASE
CDdds_POSTPAINT
CDdds_PREERASE
CDdds_PREPAINT
CDdds_SUBITEM
CDIS_CHECKED
CDIS_DEFAULT
CDIS_DISABLED
COLOR_MENUTEXT
COLOR_SCROLLBAR
COLOR_WINDOW
COLOR_WINDOWFRAME
COLOR_WINDOWTEXT
COLORMATCHTOTARGET_EMBEDED
COLORMGMTCAPS
COLORONCOLOR
COLORRES
COMCTL32_VERSION
COMPLEXREGION
CONSOLE_APPLICATION_16BIT
CONSOLE_CARET_SELECTION
CONSOLE_CARET_VISIBLE
CP_NONE
CP_RECTANGLE
CP_REGION
CREATECOLORSPACE_EMBEDED
CS_BYTEALIGNCLIENT
CS_BYTEALIGNWINDOW
CS_CLASSDC
CS_DBLCLKS
CS_DROPSHADOW
CS_GLOBALCLASS
CS_HREDRAW
CS_IME
CS_NOCLOSE
CS_OWNDC
CS_PARENTDC
CS_SAVEBITS
CS_VREDRAW
CSOUND_SYSTEM
CTLCOLOR_BTN
CTLCOLOR_DLG
CTLCOLOR_EDIT
CTLCOLOR_LISTBOX
CTLCOLOR_MAX
CTLCOLOR_MSGBOX
CTLCOLOR_SCROLLBAR
CTLCOLOR_STATIC
CURSOR_SHOWING
CURVECAPS
CWP_ALL
CWP_SKIPDISABLED
CWP_SKIPINVISIBLE
CWP_SKIPTRANSPARENT
<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA_LAST</td>
<td>DMMEDIA_GLOSSY</td>
</tr>
<tr>
<td>DC_ACTIVE</td>
<td>DMMEDIA_STANDARD</td>
</tr>
<tr>
<td>DC_BINAJUST</td>
<td>DMMEDIA_TRANSPARENCY</td>
</tr>
<tr>
<td>DC_BINNAMES</td>
<td>DMMEDIA_USER</td>
</tr>
<tr>
<td>DC_BINS</td>
<td>DMNUP_ONEUP</td>
</tr>
<tr>
<td>DC_BRUSH</td>
<td>DMNUP_SYSTEM</td>
</tr>
<tr>
<td>DC_BUTTONS</td>
<td>DMORIENT_LANDSCAPE</td>
</tr>
<tr>
<td>DC_COLLATE</td>
<td>DMORIENT_PORTRAIT</td>
</tr>
<tr>
<td>DC_COLORDEVICE</td>
<td>DMPAPER_10X11</td>
</tr>
<tr>
<td>DC_COPIES</td>
<td>DMPAPER_10X14</td>
</tr>
<tr>
<td>DC_DATATYPE_PRODUCED</td>
<td>DMPAPER_11X17</td>
</tr>
<tr>
<td>DC_DRIVER</td>
<td>DMPAPER_12X11</td>
</tr>
<tr>
<td>DC_DUPLEX</td>
<td>DMPAPER_15X11</td>
</tr>
<tr>
<td>DC_EMP_COMPLIANT</td>
<td>DMPAPER_9X11</td>
</tr>
<tr>
<td>DC_ENUMRESOLUTIONS</td>
<td>DMPAPER_A2</td>
</tr>
<tr>
<td>DC EXTRA</td>
<td>DMPAPER_A3</td>
</tr>
<tr>
<td>DC_FIELDS</td>
<td>DMPAPER_A3_EXTRA</td>
</tr>
<tr>
<td>DC_FILEDEPENDENCIES</td>
<td>DMPAPER_A3_EXTRA_TRANSVERSE</td>
</tr>
<tr>
<td>DC_GRADIENT</td>
<td>DMPAPER_A3_ROTATED</td>
</tr>
<tr>
<td>DC_HASDEFID</td>
<td>DMPAPER_A3_TRANSVERSE</td>
</tr>
<tr>
<td>DC_ICON</td>
<td>DMPAPER_A4</td>
</tr>
<tr>
<td>DC_INBUTTON</td>
<td>DMPAPER_A4_EXTRA</td>
</tr>
<tr>
<td>DC_MANUFACTURER</td>
<td>DMPAPER_A4_PLUS</td>
</tr>
<tr>
<td>DC_MAXEXTENT</td>
<td>DMPAPER_A4_ROTATED</td>
</tr>
<tr>
<td>DC_MEDIAREADY</td>
<td>DMPAPER_A4_TRANSVERSE</td>
</tr>
<tr>
<td>DC_MEDIATYPENAMES</td>
<td>DMPAPER_A4SMALL</td>
</tr>
<tr>
<td>DC_MEDIATYPES</td>
<td>DMPAPER_A5</td>
</tr>
<tr>
<td>DC_MINEXTENT</td>
<td>DMPAPER_A5_EXTRA</td>
</tr>
<tr>
<td>DC_MODEL</td>
<td>DMPAPER_A5_ROTATED</td>
</tr>
<tr>
<td>DC_NUP</td>
<td>DMPAPER_A5_TRANSVERSE</td>
</tr>
<tr>
<td>DC_ORIENTATION</td>
<td>DMPAPER_A6</td>
</tr>
<tr>
<td>DC_PAPERNAMES</td>
<td>DMPAPER_A6_ROTATED</td>
</tr>
<tr>
<td>DC_PAPERS</td>
<td>DMPAPER_A_PLUS</td>
</tr>
<tr>
<td>DC_PAPERSIZE</td>
<td>DMPAPER_B4</td>
</tr>
<tr>
<td>DC_PEN</td>
<td>DMPAPER_B4_JIS_ROTATED</td>
</tr>
<tr>
<td>DC_PERSONALITY</td>
<td>DMPAPER_B5</td>
</tr>
<tr>
<td>DC_PRINTERMEM</td>
<td>DMPAPER_B5_EXTRA</td>
</tr>
<tr>
<td>DC_PRINTRATE</td>
<td>DMPAPER_B5_JIS_ROTATED</td>
</tr>
<tr>
<td>DC_PRINTRATEPPM</td>
<td>DMPAPER_B5_TRANSVERSE</td>
</tr>
<tr>
<td>DC_PRINTRATEUNIT</td>
<td>DMPAPER_B6_JIS</td>
</tr>
<tr>
<td>DC_SIZE</td>
<td>DMPAPER_B6_JIS_ROTATED</td>
</tr>
<tr>
<td>DC_SMALLCAP</td>
<td>DMPAPER_B_PLUS</td>
</tr>
<tr>
<td>DC_STAPLE</td>
<td>DMPAPER_CSHEET</td>
</tr>
<tr>
<td>DC_TEXT</td>
<td>DMPAPER_DBL_JAPANESE_POSTCARD</td>
</tr>
</tbody>
</table>
DC_TRUETYPE DMPAPER_DBL_JAPANESE_POSTCARD_ROTATED
DC_VERSION DMPAPER_DSHEET
DCB_ACCUMULATE DMPAPER_ENV_10
DCB_DIRTY DMPAPER_ENV_11
DCB_DISABLE DMPAPER_ENV_12
DCB_ENABLE DMPAPER_ENV_14
DCB_RESET DMPAPER_ENV_9
DCB_SET DMPAPER_ENV_B4
DCBA_FACEDOWNCENTER DMPAPER_ENV_B5
DCBA_FACEDOWNLEFT DMPAPER_ENV_B6
DCBA_FACEDOWNNONE DMPAPER_ENV_C3
DCBA_FACEDOWNRIGHT DMPAPER_ENV_C4
DCBA_FACEUPCENTER DMPAPER_ENV_C5
DCBA_FACEUPLEFT DMPAPER_ENV_C6
DCBA_FACEUPNONE DMPAPER_ENV_C65
DCBA_FACEUPRIGHT DMPAPER_ENV_DL
DC_CACHE DMPAPER_ENV_INVITE
DC_CLIPCHILDREN DMPAPER_ENV_ITALY
DC_CLIPSLIBINGS DMPAPER_ENV_MONARCH
DC_EXCLUDERGN DMPAPER_ENV_PERSONAL
DC_EXCLUDEUPDATE DMPAPER_ESHEET
DC_INTERSECTRGN DMPAPER_EXECUTIVE
DC_INTERSECTUPDATE DMPAPER_FANFOLD_LGL_GERMAN
DC_LOCKWINDOWUPDATE DMPAPER_FANFOLD_STD_GERMAN
DC_NORESETATTRS DMPAPER_FANFOLD_US
DC_PARENTCLIP DMPAPER_FOLIO
DC_VALIDATE DMPAPER_ISO_B4
DC_WINDOW DMPAPER_JAPANESE_POSTCARD
DDL_ARCHIVE DMPAPER_JAPANESE_POSTCARD_ROTATED
DDL_DIRECTORY DMPAPER_JENV_CHOU3
DDL_DRIVES DMPAPER_JENV_CHOU3_ROTATED
DDL_EXCLUSIVE DMPAPER_JENV_CHOU4
DDL_HIDDEN DMPAPER_JENV_CHOU4_ROTATED
DDL_POSTMSG DMPAPER_JENV_KAKU2
DDL_READONLY DMPAPER_JENV_KAKU2_ROTATED
DDL_READWRITE DMPAPER_JENV_KAKU3
DDL_SYSTEM DMPAPER_JENV_KAKU3_ROTATED
DEFAULT_CHARSET DMPAPER_JENV_YOU4
DEFAULT_GUI_FONT DMPAPER_JENV_YOU4_ROTATED
DEFAULT_PALETTE DMPAPER_LAST
DEFAULT_PITCH DMPAPER_LEDGER
DEFAULT_QUALITY DMPAPERLEGAL
DESKTOP_CREATEMENU DMPAPERLEGAL_EXTRA
DESKTOP_CREATEWINDOW DMPAPER_LETTER
DESKTOP_ENUMERATE DMPAPER_LETTER_EXTRA
DESKTOP_HOOKCONTROL DMPAPER_LETTER_EXTRA_TRANSVERSE
DESKTOP_JOURNALPLAYBACK DMPAPER_LETTER_PlUS
DESKTOP_JOURNALRECORD DMPAPER_LETTER_ROTATED
DESKTOP_READOBJECTS DMPAPER_LETTER_TRANSVERSE
DESKTOP_SWITCHDESKTOP DMPAPER_LETTERSMALL
DESKTOP_WRITEOBJECTS DMPAPER_NOTE
DESKTOPHORZRES DMPAPER_P16K
DESKTOPVERTRES DMPAPER_P16K_ROTATED
DEVICE_DEFAULT_FONT DMPAPER_P32K
DEVICE_FONTTYPE DMPAPER_P32K_ROTATED
DEVICE_NOTIFY_ALL_INTERFACE_CLASSES DMPAPER_P32KBIG
DEVICE_NOTIFY_SERVICE_HANDLE DMPAPER_P32KBIG_ROTATED
DEVICE_NOTIFY_WINDOW_HANDLE DMPAPER_PENV_1
DEVICEDATA DMPAPER_PENV_10
DFC_BUTTON DMPAPER_PENV_10_ROTATED
DFC_CAPTION DMPAPER_PENV_1_ROTATED
DFC_MENU DMPAPER_PENV_2
DFCPopover DMPAPER_PENV_2_ROTATED
DFC_SCROLL DMPAPER_PENV_3
DFC_ADJUSTRECT DMPAPER_PENV_3_ROTATED
DFCS_BUTTONSTATE DMPAPER_PENV_4
DFCS_BUTTONCHECK DMPAPER_PENV_4_ROTATED
DFCS_BUTTONPUSH DMPAPER_PENV_5
DFCS_BUTTONRADIO DMPAPER_PENV_5_ROTATED
DFCS_BUTTONRADIOIMAGE DMPAPER_PENV_6
DFCS_BUTTONRADIOMASK DMPAPER_PENV_6_ROTATED
DFCS_CAPTIONCLOSE DMPAPER_PENV_7
DFCS_CAPTIONHELP DMPAPER_PENV_7_ROTATED
DFCS_CAPTIONMAX DMPAPER_PENV_8
DFCS_CAPTIONMIN DMPAPER_PENV_8_ROTATED
DFCS_CAPTIONRESTORE DMPAPER_PENV_9
DFCS_CHECKED DMPAPER_PENV_9_ROTATED
DFCS_FLAT DMPAPER_QUARTO
DFCS_HOT DMPAPER_RESERVED_48
DFCS_INACTIVE DMPAPER_RESERVED_49
DFCS_MENUARROW DMPAPER_STATEMENT
DFCS_MENUARROWRIGHT DMPAPER_TABLOID
DFCS_MENUBULLET DMPAPER_TABLOID_EXTRA
DFCS_MENUCHECK DMPAPER_USER
DFCS_MONO DMRES_DRAFT
DFCS_PUSHED DMRES_HIGH
DFCS_SCROLLCOMBOBOX DMRES_LOW
DFCS_SCROLLDOWN DMRES_MEDIUM
DFCS_SCROLLLEFT DMTT_BITMAP
DFCS_SCROLLRIGHT DMTT_DOWNLOAD
DFCS_SCROLLSIZEGRIP DMTT_DOWNLOAD_OUTLINE
DFCS_SCROLLSIZEGRIPRIGHT DMTT_SUBDEV
DFCS_SCROLLUP
DFCS_TRANSPARENT
DI_APPBANDING
DI_COMPAT
DI_DEFAULTSIZE
DI_IMAGE
DI_MASK
DI_NOMIRROR
DI_NORMAL
DI_ROPS_READ_DESTINATION
DIB_PAL_COLORS
DIB_RGB_COLORS
DIFFERENCE
DISP_CHANGE_BADDUALVIEW
DISP_CHANGE_BADFLAGS
DISP_CHANGE_BADMODE
DISP_CHANGE_BADPARAM
DISP_CHANGE_FAILED
DISP_CHANGE_NOTUPDATED
DISP_CHANGE_RESTART
DISP_CHANGE_SUCCESSFUL
DISPLAY_DEVICE_ACTIVE
DISPLAY_DEVICE_ATTACHED
DISPLAY_DEVICE_ATTACHED_TO_DESKTOP
DISPLAY_DEVICE_DISCONNECT
DISPLAY_DEVICE_MIRRORING_DRIVER
DISPLAY_DEVICE_MODESPRUNED
DISPLAY_DEVICE_MULTI_DRIVER
DISPLAY_DEVICE_PRIMARY_DEVICE
DISPLAY_DEVICE_REMOTE
DISPLAY_DEVICE_REMOVABLE
DISPLAY_DEVICE_VGA_COMPATIBLE
DKGRAY_BRUSH
DL_BEGINDRAG
DL_CANCELDrag
DL_COPYCURSOR
DL_CURSORSET
DL_CURSORSET
DL_DRAGGING
DL_DROPPED
DL_MOVECURSOR
DL_STOPCURSOR
DLGC_BUTTON
DLGC_DEFPUSHBUTTON
DLGC_HASSETSEL
DLGC_RADIOBUTTON
DLGC_STATIC
<table>
<thead>
<tr>
<th>Constant</th>
<th>Associated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLGC_UNDEFPUSHBUTTON</td>
<td>DST_COMPLEX</td>
</tr>
<tr>
<td>DLGC_WANTALLKEYS</td>
<td>DST_ICON</td>
</tr>
<tr>
<td>DLGC_WANTARROWS</td>
<td>DST_PREFIXTEXT</td>
</tr>
<tr>
<td>DLGC_WANTCHARS</td>
<td>DST_TEXT</td>
</tr>
<tr>
<td>DLGC_WANTMESSAGE</td>
<td>DT_BOTTOM</td>
</tr>
<tr>
<td>DLGC_WANTTAB</td>
<td>DT_CALCRECT</td>
</tr>
<tr>
<td>DLGWINDOWEXTRA</td>
<td>DT_CENTER</td>
</tr>
<tr>
<td>DLSZ_MOVE_X</td>
<td>DT_CHARSSTREAM</td>
</tr>
<tr>
<td>DLSZ_MOVE_Y</td>
<td>DT_DISPFILE</td>
</tr>
<tr>
<td>DLSZ_REPAINT</td>
<td>DT_EDITCONTROL</td>
</tr>
<tr>
<td>DLSZ_SIZE_X</td>
<td>DT_END_ELLIPSIS</td>
</tr>
<tr>
<td>DLSZ_SIZE_Y</td>
<td>DT_EXPANDTAB</td>
</tr>
<tr>
<td>DM_COPY</td>
<td>DT_EXTERNALLEADING</td>
</tr>
<tr>
<td>DM_GETDEFID</td>
<td>DT_HIDEPREFIX</td>
</tr>
<tr>
<td>DM_IN_BUFFER</td>
<td>DT_INTERNAL</td>
</tr>
<tr>
<td>DM_IN_PROMPT</td>
<td>DT_LEFT</td>
</tr>
<tr>
<td>DM_MODIFY</td>
<td>DT_METAFILE</td>
</tr>
<tr>
<td>DM_OUT_BUFFER</td>
<td>DT_MODIFYSTRING</td>
</tr>
<tr>
<td>DM_OUT_DEFAULT</td>
<td>DT_NOCLIP</td>
</tr>
<tr>
<td>DM_PROMPT</td>
<td>DT_NOFULLWIDTHCHARBREAK</td>
</tr>
<tr>
<td>DM_REPOSITION</td>
<td>DT_NOPREFIX</td>
</tr>
<tr>
<td>DM_SETDEFID</td>
<td>DT_PATH_ELLIPSIS</td>
</tr>
<tr>
<td>DM_SPECVERSION</td>
<td>DT_PLOTTER</td>
</tr>
<tr>
<td>DM_UPDATE</td>
<td>DT_PREFIXONLY</td>
</tr>
<tr>
<td>DMBIN_AUTO</td>
<td>DT_RASCAMERA</td>
</tr>
<tr>
<td>DMBIN_CASSETTE</td>
<td>DT_RASDISPLAY</td>
</tr>
<tr>
<td>DMBIN_ENVELOPE</td>
<td>DT_RASPRINTER</td>
</tr>
<tr>
<td>DMBIN_ENVMANUAL</td>
<td>DT_RIGHT</td>
</tr>
<tr>
<td>DMBIN_FORMSOURCE</td>
<td>DT_RTLREADING</td>
</tr>
<tr>
<td>DMBIN_LARGECAPACITY</td>
<td>DT_SINGLELINE</td>
</tr>
<tr>
<td>DMBIN_LARGEFMT</td>
<td>DT_TABSTOP</td>
</tr>
<tr>
<td>DMBIN_LAST</td>
<td>DT_TOP</td>
</tr>
<tr>
<td>DMBIN_LOWER</td>
<td>DT_VCENTER</td>
</tr>
<tr>
<td>DMBIN_MANUAL</td>
<td>DT_WORD_ELLIPSIS</td>
</tr>
<tr>
<td>DMBIN_MIDDLE</td>
<td>DT_WORDBREAK</td>
</tr>
<tr>
<td>DMBIN_ONLYONE</td>
<td>DTM_FIRST</td>
</tr>
<tr>
<td>DMBIN_SMALLFMT</td>
<td>DTM_GETMCCOLOR</td>
</tr>
<tr>
<td>DMBINTRACTOR</td>
<td>DTM_GETMCFONT</td>
</tr>
<tr>
<td>DMBIN_UPPER</td>
<td>DTM_GETMONTHCAL</td>
</tr>
<tr>
<td>DMBIN_USER</td>
<td>DTM_GETRANGE</td>
</tr>
<tr>
<td>DMCOLLATE_FALSE</td>
<td>DTM_GETSYSTEMTIME</td>
</tr>
<tr>
<td>DMCOLLATE_TRUE</td>
<td>DTM_SETFORMAT</td>
</tr>
<tr>
<td>DMCOLOR_COLOR</td>
<td>DTM_SETFORMATA</td>
</tr>
<tr>
<td>DMCOLOR_MONOCHROME</td>
<td>DTM_SETFORMATW</td>
</tr>
<tr>
<td>DMDFO_CENTER</td>
<td>DTM_SETMCCOLOR</td>
</tr>
<tr>
<td>DMDFO_DEFAULT</td>
<td>DTM_SETMCFONT</td>
</tr>
<tr>
<td>Constants</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>DMDFO_STRETCH</td>
<td>DTM_SETRANGE</td>
</tr>
<tr>
<td>DMDisplayFlags_TextMode</td>
<td>DTM_SETSYSTEMTIME</td>
</tr>
<tr>
<td>DMDither_Coarse</td>
<td>DTM_CLOSEUP</td>
</tr>
<tr>
<td>DMDither_ErrorDiffusion</td>
<td>DTM_DATETIMECHANGE</td>
</tr>
<tr>
<td>DMDither_Fine</td>
<td>DTM_DROPDOWN</td>
</tr>
<tr>
<td>DMDither_GrayScale</td>
<td>DTM_FORMAT</td>
</tr>
<tr>
<td>DMDither_LineArt</td>
<td>DTM_FORMATA</td>
</tr>
<tr>
<td>DMDither_None</td>
<td>DTM_FORMATQUERY</td>
</tr>
<tr>
<td>DMDither_RESERVED6</td>
<td>DTM_FORMATQUERYA</td>
</tr>
<tr>
<td>DMDither_RESERVED7</td>
<td>DTM_FORMATQUERYW</td>
</tr>
<tr>
<td>DMDither_RESERVED8</td>
<td>DTN_FORMAT</td>
</tr>
<tr>
<td>DMDither_RESERVED9</td>
<td>DTN_USERSTRING</td>
</tr>
<tr>
<td>DMDither_USER</td>
<td>DTN_USERSTRINGA</td>
</tr>
<tr>
<td>DMDO_180</td>
<td>DTN_USERSTRINGW</td>
</tr>
<tr>
<td>DMDO_270</td>
<td>DTM_WMKEYDOWN</td>
</tr>
<tr>
<td>DMDO_90</td>
<td>DTM_WMKEYDOWNA</td>
</tr>
<tr>
<td>DMDO_DEFAULT</td>
<td>DTM_WMKEYDOWNW</td>
</tr>
<tr>
<td>DMDup_Horizontal</td>
<td>DTS_APPCANPARSE</td>
</tr>
<tr>
<td>DMDup_Simplex</td>
<td>DTS_LONGDATEFORMAT</td>
</tr>
<tr>
<td>DMDup_Vertical</td>
<td>DTS_RIGHTALIGN</td>
</tr>
<tr>
<td>DMICM_ABS_Colormetric</td>
<td>DTS_SHORTDATECENTURYFORMAT</td>
</tr>
<tr>
<td>DMICM_Colormetric</td>
<td>DTS_SHORTDATEFORMAT</td>
</tr>
<tr>
<td>DMICM_Contrast</td>
<td>DTS_SHOWNONE</td>
</tr>
<tr>
<td>DMICM_Saturate</td>
<td>DTS_TIMEFORMAT</td>
</tr>
<tr>
<td>DMICM_User</td>
<td>DTS_UPDOWN</td>
</tr>
<tr>
<td>DMICMMethod_Device</td>
<td>DWL_DLGPROC</td>
</tr>
<tr>
<td>DMICMMethod_Driver</td>
<td>DWL_MSGRESULT</td>
</tr>
<tr>
<td>DMICMMethod_None</td>
<td>DWL_USER</td>
</tr>
<tr>
<td>DMICMMethod_System</td>
<td>DWLP_MSGRESULT</td>
</tr>
<tr>
<td>DMICMMethod_User</td>
<td></td>
</tr>
</tbody>
</table>

**E**

<table>
<thead>
<tr>
<th>Constants</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EASTEUROPE_CHARSET</td>
<td>EMR_SETARCDIRECTION</td>
</tr>
<tr>
<td>EC_LEFTMARGIN</td>
<td>EMR_SETBKCOLOR</td>
</tr>
<tr>
<td>EC_RIGHTMARGIN</td>
<td>EMR_SETBKMODE</td>
</tr>
<tr>
<td>EC_USEFONTINFO</td>
<td>EMR_SETBRUSHORGEX</td>
</tr>
<tr>
<td>ECM_FIRST</td>
<td>EMR_SETCOLORADJUSTMENT</td>
</tr>
<tr>
<td>EDGE_Bump</td>
<td>EMR_SETCOLORSPACE</td>
</tr>
<tr>
<td>EDGE_Etched</td>
<td>EMR_SETDIBITSTODEVICE</td>
</tr>
<tr>
<td>EDGE_Raised</td>
<td>EMR_SETICMMODE</td>
</tr>
<tr>
<td>EDGE_Sunken</td>
<td>EMR_SETICMPROFILEA</td>
</tr>
<tr>
<td>EIMES_CancelCompStrInfoFocus</td>
<td>EMR_SETICMPROFILEW</td>
</tr>
<tr>
<td>EIMES_CompleteCompStrKillFocus</td>
<td>EMR_SETLAYOUT</td>
</tr>
<tr>
<td>EIMES_GetCompStratonce</td>
<td>EMR_SETMAPMODE</td>
</tr>
<tr>
<td>ELF_Culture_Latin</td>
<td>EMR_SETMAPPERFLAGS</td>
</tr>
<tr>
<td>Constants</td>
<td>EMR/Set*</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>ELF_VENDOR_SIZE</td>
<td>EMR_SETMETARGN</td>
</tr>
<tr>
<td>ELF_VERSION</td>
<td>EMR_SETMETERLIMIT</td>
</tr>
<tr>
<td>EM_CANUNDO</td>
<td>EMR_SETPALETTEENTRIES</td>
</tr>
<tr>
<td>EM_CHARFROMPOS</td>
<td>EMR_SETPIXELV</td>
</tr>
<tr>
<td>EM_EMPTYUNDOBUFFER</td>
<td>EMR_SETPOLYFILLMODE</td>
</tr>
<tr>
<td>EM_FMTLINES</td>
<td>EMR_SETROP2</td>
</tr>
<tr>
<td>EM_GETCUEBANNER</td>
<td>EMR_SETSTRETCHBLTMODE</td>
</tr>
<tr>
<td>EM_GETFIRSTVISIBLELINE</td>
<td>EMR_SETTEXALIGN</td>
</tr>
<tr>
<td>EM_GETHANDLE</td>
<td>EMR_SETTEXTCOLOR</td>
</tr>
<tr>
<td>EM_GETTIMESTATUS</td>
<td>EMR_SETVIEWPORTTEXTEX</td>
</tr>
<tr>
<td>EM_GETLIMITTEXT</td>
<td>EMR_SETVIEWPORTORGEX</td>
</tr>
<tr>
<td>EM_GETLINE</td>
<td>EMR_SETWINDOWTEXT</td>
</tr>
<tr>
<td>EM_GETLINECOUNT</td>
<td>EMR_SETWINDOWORGEX</td>
</tr>
<tr>
<td>EM_GETMARGINS</td>
<td>EMR_SETWORLDTRANSFORM</td>
</tr>
<tr>
<td>EM_GETMODIFY</td>
<td>EMR_STRETCHBLT</td>
</tr>
<tr>
<td>EM_GETPASSWORDCHAR</td>
<td>EMR_STRETCHDIBITS</td>
</tr>
<tr>
<td>EM_GETRECT</td>
<td>EMR_STROKEDANDFILLPATH</td>
</tr>
<tr>
<td>EM_GETSEL</td>
<td>EMR_STROKEPATH</td>
</tr>
<tr>
<td>EM_GETTHUMB</td>
<td>EMR_TRANSPARENTBLT</td>
</tr>
<tr>
<td>EM_GETWORDBREAKPROC</td>
<td>EMR_WIDENPATH</td>
</tr>
<tr>
<td>EM_HIDEBALLOONTIP</td>
<td>EMSIS_COMPOSITIONSTRING</td>
</tr>
<tr>
<td>EM_LIMITTEXT</td>
<td>EN_ALIGN_LTR_EC</td>
</tr>
<tr>
<td>EM_LINEFROMCHAR</td>
<td>EN_ALIGN_RTL_EC</td>
</tr>
<tr>
<td>EM_LINEINDEX</td>
<td>EN_CHANGE</td>
</tr>
<tr>
<td>EM_LINELENGTH</td>
<td>EN_ERRSPACE</td>
</tr>
<tr>
<td>EM_LINESCROLL</td>
<td>EN_HSCROLL</td>
</tr>
<tr>
<td>EM_POSFROMCHAR</td>
<td>EN_KILLFOCUS</td>
</tr>
<tr>
<td>EM_REPLACESELE</td>
<td>EN_MAXTEXT</td>
</tr>
<tr>
<td>EM_SCROLL</td>
<td>EN_SETFOCUS</td>
</tr>
<tr>
<td>EM_SCROLLCARET</td>
<td>EN_UPDATE</td>
</tr>
<tr>
<td>EM_SETCUEBANNER</td>
<td>EN_VSCROLL</td>
</tr>
<tr>
<td>EM_SETHANDLE</td>
<td>ENABLEDUPLEX</td>
</tr>
<tr>
<td>EM_SETTIMESTATUS</td>
<td>ENABLEPAIRKERNING</td>
</tr>
<tr>
<td>EM_SETLIMITTEXT</td>
<td>ENABLERELATIVEWIDTHS</td>
</tr>
<tr>
<td>EM_SETMARGINS</td>
<td>ENCAPSULATED_POSTSCRIPT</td>
</tr>
<tr>
<td>EM_SETMODIFY</td>
<td>END_PATH</td>
</tr>
<tr>
<td>EM_SETPASSWORDCHAR</td>
<td>ENDDOC</td>
</tr>
<tr>
<td>EM_SETREADONLY</td>
<td>ENDSESSION_LOGOFF</td>
</tr>
<tr>
<td>EM_SETRECT</td>
<td>ENHMETA_SIGNATURE</td>
</tr>
<tr>
<td>EM_SETRECTNP</td>
<td>ENHMETA_STOCK_OBJECT</td>
</tr>
<tr>
<td>EM_SETSEL</td>
<td>ENUMPAPERBINS</td>
</tr>
<tr>
<td>EM_SETTABSTOPS</td>
<td>ENUMPAPERMETRICS</td>
</tr>
<tr>
<td>EM_SETWORDBREAKPROC</td>
<td>EPS_SIGNATURE</td>
</tr>
<tr>
<td>EM_SHOWBALLOONTIP</td>
<td>EPSPRINTING</td>
</tr>
<tr>
<td>EM_UNDO</td>
<td>ERROR</td>
</tr>
<tr>
<td>EMR_ABORTPATH</td>
<td>ES_AUTOHSCROLL</td>
</tr>
<tr>
<td>EMR_OFFSETCLIPRGN</td>
<td>EVENT_OBJECT_LOCATIONCHANGE</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>EMR_PAINTRGN</td>
<td>EVENT_OBJECT_NAMECHANGE</td>
</tr>
<tr>
<td>EMR_PIE</td>
<td>EVENT_OBJECT_PARENTCHANGE</td>
</tr>
<tr>
<td>EMR_PIXELFORMAT</td>
<td>EVENT_OBJECT_REORDER</td>
</tr>
<tr>
<td>EMR_PLINE</td>
<td>EVENT_OBJECT_SELECTION</td>
</tr>
<tr>
<td>EMR_PLINE16</td>
<td>EVENT_OBJECT_SELECTIONADD</td>
</tr>
<tr>
<td>EMR_PLINE1616</td>
<td>EVENT_OBJECT_SELECTIONREMOVE</td>
</tr>
<tr>
<td>EMR_PLINE161616</td>
<td>EVENT_OBJECT_SELECTIONWITHIN</td>
</tr>
<tr>
<td>EMR_PLINE16161616</td>
<td>EVENT_OBJECT_SHOW</td>
</tr>
<tr>
<td>EMR_PLINE1616161616</td>
<td>EVENT_OBJECT_STATECHANGE</td>
</tr>
<tr>
<td>EMR_PLINE161616161616</td>
<td>EVENT_OBJECT_VALUECHANGE</td>
</tr>
<tr>
<td>EMR_POLYGON</td>
<td>EVENT_SYSTEM_ALERT</td>
</tr>
<tr>
<td>EMR_POLYGON16</td>
<td>EVENT_SYSTEM_CAPTUREEEND</td>
</tr>
<tr>
<td>EMR_POLYLINE</td>
<td>EVENT_SYSTEM_CAPTURESTART</td>
</tr>
<tr>
<td>EMR_POLYLINE16</td>
<td>EVENT_SYSTEM_CONTEXTHELPEND</td>
</tr>
<tr>
<td>EMR_POLYLINE1616</td>
<td>EVENT_SYSTEMCONTEXTHELPSTART</td>
</tr>
<tr>
<td>EMR_POLYLINE161616</td>
<td>EVENT_SYSTEM_DIALOGEND</td>
</tr>
<tr>
<td>EMR_POLYLINE1616161616</td>
<td>EVENT_SYSTEM_DIALOGSTART</td>
</tr>
<tr>
<td>EMR_POLYLINE161616161616</td>
<td>EVENT_SYSTEM_DROGDROPEND</td>
</tr>
<tr>
<td>EMR_POLYLINE16161616161616</td>
<td>EVENT_SYSTEM_DROGDROPSTART</td>
</tr>
<tr>
<td>EMR_POLYLINE1616161616161616</td>
<td>EVENT_SYSTEM_FOREGROUND</td>
</tr>
<tr>
<td>EMR_POLYTEXTOUTA</td>
<td>EVENT_SYSTEM_MENUEND</td>
</tr>
<tr>
<td>EMR_POLYTEXTOUTW</td>
<td>EVENT_SYSTEM_MENUPopupEND</td>
</tr>
<tr>
<td>EMR_REALIZEPALETTE</td>
<td>EVENT_SYSTEM_MENUPopupSTART</td>
</tr>
<tr>
<td>EMR_RECTANGLE</td>
<td>EVENT_SYSTEM_MENUSTART</td>
</tr>
<tr>
<td>EMR_RESERVED_105</td>
<td>EVENT_SYSTEM_MINIMIZEEEND</td>
</tr>
<tr>
<td>EMR_RESERVED_106</td>
<td>EVENT_SYSTEM_MINIMIZESTART</td>
</tr>
<tr>
<td>EMR_RESERVED_107</td>
<td>EVENT_SYSTEM_MOVESIZEEEND</td>
</tr>
<tr>
<td>EMR_RESERVED_108</td>
<td>EVENT_SYSTEM_MOVESIZESTART</td>
</tr>
<tr>
<td>EMR_RESERVED_109</td>
<td>EVENT_SYSTEM_SCROLLINGEEND</td>
</tr>
<tr>
<td>EMR_RESERVED_110</td>
<td>EVENT_SYSTEM_SCROLLINGSTART</td>
</tr>
<tr>
<td>EMR_RESERVED_117</td>
<td>EVENT_SYSTEM_SOUND</td>
</tr>
<tr>
<td>EMR_RESERVED_119</td>
<td>EVENT_SYSTEM_SWITCHEEND</td>
</tr>
<tr>
<td>EMR_RESERVED_120</td>
<td>EVENT_SYSTEM_SWITCHSTART</td>
</tr>
<tr>
<td>EMR_RESIZEPALETTE</td>
<td>EWX_FORCE</td>
</tr>
<tr>
<td>EMR_RESTOREDC</td>
<td>EWX_FORCEIFHUNG</td>
</tr>
<tr>
<td>EMR_ROUNDRECT</td>
<td>EWX_LOGOFF</td>
</tr>
<tr>
<td>EMR_SAVEDC</td>
<td>EWX_POWEROFF</td>
</tr>
<tr>
<td>EMR_SCALEVIEWPORTEXTEX</td>
<td>EWX_REBOOT</td>
</tr>
<tr>
<td>EMR_SCALEWINDOWEXTEX</td>
<td>EWX_SHUTDOWN</td>
</tr>
<tr>
<td>EMR_SELECTCLIPPATH</td>
<td>EXT_DEVICE_CAPS</td>
</tr>
<tr>
<td>EMR_SELECTOBJECT</td>
<td>EXTEXTOUT</td>
</tr>
<tr>
<td>EMR_SELECTPALETTE</td>
<td></td>
</tr>
</tbody>
</table>
F

FALSE
FALT
FAPPCOMMAND_KEY
FAPPCOMMAND_MASK
FAPPCOMMAND_MOUSE
FAPPCOMMAND_OEM
FCONTROL
FE_FONTSMOOTHINGCLEARTYPE
FE_FONTSMOOTHINGDOCKING
FE_FONTSMOOTHINGORIENTATIONBGR
FE_FONTSMOOTHINGORIENTATIONRGB
FE_FONTSMOOTHINGSTANDARD
FEATURESETTING_CUSTPAPER
FEATURESETTING_MIRROR
FEATURESETTING_NEGATIVE
FEATURESETTING_NUP
FEATURESETTING_OUTPUT
FEATURESETTING_PRIVATE_BEGIN
FEATURESETTING_PRIVATE_END
FEATURESETTING_PROTOCOL
FEATURESETTING_PSLEVEL
FF_DECORATIVE
FF_DONTCARE
FF_MODERN
FF_ROMAN
FF_SCRIPT
FF_SWISS
FIXED_PITCH
FKF_AVAILABLE
FKF_CLICKON
FKF_CONFIRMHOTKEY
FKF_FILTERKEYSON
FKF_HOTKEYACTIVE
FKF_HOTKEYSOUND
FKF_INDICATOR
FLASHW_ALL
FLASHW_CAPTION
FLASHW_STOP
FLASHW_TIMER
FLASHW_TIMERNOFG
FLASHW_TRAY
FLI_MASK
FLOODFILLBORDER
FLOODFILLSURFACE
FNOINVERT
FONTMAPPERR_MAX
FR_DIALOGTERM
FR_DOWN
FR_ENABLEHOOK
FR_ENABLETEMPLATE
FR_ENABLETEMPLATEHANDLE
FR_FINDNEXT
FR_HIDEMATCHCASE
FR_HIDEUPDOWN
FR_HIDEWHOLEWORD
FR_MATCHALEFHAMZA
FR_MATCHCASE
FR_MATCHDIAC
FR.MatchKashida
FR.NOMATCHCASE
FR_NOT_ENUM
FR_NOWHOLEWORD
FR_PRIVATE
FR_RAW
FR_REPLACE
FR_REPLACEALL
FR_SHOWHELP
FR.WHOLEWORD
FSB_ENCARTA_MODE
FSB_FLAT_MODE
FSB_REGULAR_MODE
FSHIFT
FVIRTKEY
FW_BLACK
FW_BOLD
FW_DEMIBOLD
FW_DONTCARE
FW_EXTRABOLD
FW_EXTRALIGHT
FW_HEAVY
FW_LIGHT
FW_MEDIUM
FW_NORMAL
FW_REGULAR
FW_SEMIBOLD
FW_THIN
FW_ULTRABOLD
<table>
<thead>
<tr>
<th>Constants</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUSHOUTPUT</td>
<td>FW_ULTRALIGHT</td>
</tr>
<tr>
<td>GA_PARENT</td>
<td>GETPRINTINGOFFSET</td>
</tr>
<tr>
<td>GA_ROOT</td>
<td>GETSCALINGFACTOR</td>
</tr>
<tr>
<td>GA_ROOTOWNER</td>
<td>GETSETPAPERBINS</td>
</tr>
<tr>
<td>GB2312_CHARSET</td>
<td>GETSETPAPERMETRICS</td>
</tr>
<tr>
<td>GCL_CBCLSEXTRA</td>
<td>GETSETPRINTORIENT</td>
</tr>
<tr>
<td>GCL_CBWNDSEXTRA</td>
<td>GETSETSCREENPARAMS</td>
</tr>
<tr>
<td>GCL_HBRBACKGROUND</td>
<td>GETTECHNOLOGY</td>
</tr>
<tr>
<td>GCL_HCURSOR</td>
<td>GETTECHNOLOGY</td>
</tr>
<tr>
<td>GCL_HICON</td>
<td>GETTRACKKERNTABLE</td>
</tr>
<tr>
<td>GCL_HICONSM</td>
<td>GETVECTORBRUSHSIZE</td>
</tr>
<tr>
<td>GCL_HMODULE</td>
<td>GETVECTORPENSIZE</td>
</tr>
<tr>
<td>GCL_MENUNAME</td>
<td>GGI_MARK_NONEXISTING_GLYPHS</td>
</tr>
<tr>
<td>GCL_STYLE</td>
<td>GGO_BEZIER</td>
</tr>
<tr>
<td>GCL_WNDPROC</td>
<td>GGO_BITMAP</td>
</tr>
<tr>
<td>GCLP_HBRBACKGROUND</td>
<td>GGO_GLYPH_INDEX</td>
</tr>
<tr>
<td>GCLP_HCURSOR</td>
<td>GGO_GRAY2_BITMAP</td>
</tr>
<tr>
<td>GCLP_HICON</td>
<td>GGO_GRAY4_BITMAP</td>
</tr>
<tr>
<td>GCLP_HICONSM</td>
<td>GGO_GRAY8_BITMAP</td>
</tr>
<tr>
<td>GCLP_HMODULE</td>
<td>GGO_METRICS</td>
</tr>
<tr>
<td>GCLP_MENUNAME</td>
<td>GGO_NATIVE</td>
</tr>
<tr>
<td>GCLP_WNDPROC</td>
<td>GGO_UNHINTED</td>
</tr>
<tr>
<td>GCP_DBCS</td>
<td>GM_ADVANCED</td>
</tr>
<tr>
<td>GCP_DIACRITIC</td>
<td>GM_COMPATIBLE</td>
</tr>
<tr>
<td>GCP_ERROR</td>
<td>GM_LAST</td>
</tr>
<tr>
<td>GCP_GLYPHSHAPE</td>
<td>GMDI_GOINTOPOPUPS</td>
</tr>
<tr>
<td>GCP_KASHIDA</td>
<td>GMDI_USEDISABLE</td>
</tr>
<tr>
<td>GCP_LIGATE</td>
<td>GMMP_USE_DISPLAY_POINTS</td>
</tr>
<tr>
<td>GCP_REORDER</td>
<td>GMMP_USE_HIGH_RESOLUTION_POINTS</td>
</tr>
<tr>
<td>GCP_USEKERNING</td>
<td>GMR_DAYSTATE</td>
</tr>
<tr>
<td>GCPCLASS_ARABIC</td>
<td>GR_GDIOBJECTS</td>
</tr>
<tr>
<td>GCPCLASS_HEBREW</td>
<td>GR_USEROBJECTS</td>
</tr>
<tr>
<td>GCPCLASS_LATIN</td>
<td>GR_USEROBJECTS</td>
</tr>
<tr>
<td>GCPCLASS_LATINNUMBER</td>
<td>GRADIENT_FILL_OP_FLAG</td>
</tr>
<tr>
<td>GCPCLASS_LATINNUMERICSEPARATOR</td>
<td>GRADIENT_FILL_RECT_H</td>
</tr>
<tr>
<td>GCPCLASS_LATINNUMERICTERMINATOR</td>
<td>GRADIENT_FILL_RECT_V</td>
</tr>
<tr>
<td>GCPCLASS_LOCALNUMBER</td>
<td>GRADIENT_FILL_TRIANGLE</td>
</tr>
<tr>
<td>GCPCLASS_NEUTRAL</td>
<td>GRAY_BRUSH</td>
</tr>
<tr>
<td>GCPCLASS_NUMERICSEPARATOR</td>
<td>GREEK_CHARSET</td>
</tr>
<tr>
<td>GCPCLASS_POSTBOUNDLTR</td>
<td>GS_8BIT_INDICES</td>
</tr>
<tr>
<td>GCPCLASS_POSTBOUNDRTL</td>
<td>GUI_16BITTASK</td>
</tr>
<tr>
<td>GCPCLASS_PREBOUNDLTR</td>
<td>GUI_CARETBLINKING</td>
</tr>
<tr>
<td>GCPCLASS_PREBOUNDRTL</td>
<td>GUI_INMENUMODE</td>
</tr>
</tbody>
</table>
GCPGLYPH_LINKAFTER GUI_INMOVESIZE
GCPGLYPH_LINKBEFORE GUI_POPUPMENUMODE
GCW_ATOM GUI_SYSTEMMENUMODE
GDICOMMENT_BEGINGROUP GW_CHILD
GDICOMMENT_ENDGROUP GW_ENABLEDPOPUP
GDICOMMENT_IDENTIFIER GW_HWNDFIRST
GDICOMMENT_MULTIFORMATS GW_HWNDLAST
GDICOMMENT_UNICODE_STRING GW_HWNDNEXT
GDICOMMENT_WINDOWS_METAFILE GW_HWNDPREV
GDT_ERROR GW_OWNER
GDT_NONE GWL_EXSTYLE
GDT_VALID GWL_HINSTANCE
GDTR_MAX GWL_HWNDPARENT
GDTR_MIN GWL_ID
GET_PS_FEATURESETTING GWL_STYLE
GETCOLORABLE GWL_USERDATA
GETDEVICEUNITS GWL_WNDPROC
GETEXTENDEDTEXTMETRICS GWLP_HINSTANCE
GETEXTENTTABLE GWLP_HWNDPARENT
GETFACENAME GWLP_ID
GETFAIRKERNTABLE GWLP_USERDATA
GETPENWIDTH GWLP_WNDPROC
GETPHYSFEATURECONTEXT

H
HALFTONE HELP_CONTENTS
HANGEUL_CHARSET HELP_CONTEXT
HANGUL_CHARSET HELP_CONTEXTMENU
HC_ACTION HELP_FINDER
HC_GETNEXT HELP_FORCEFILE
HC_NOREM HELP_INDEX
HC_NOREMOVE HELP_HELPONHELP
HC_SKIP HELP_KEY
HC_SYSMODALOFF HELP_MULTIKEY
HC_SYSMODALON HELP_PARTIALKEY
HCBT_ACTIVATE HELP_QS
HCBT_CLICKSKIPPED HELP_TCARD
HCBT_CREATEWND HELP_TCARD_DATA
HCBT_DESTROYWND HELP_TCARD_OTHER_CALLER
HCBT_KEYSKIPPED HELP_TCARD
HCBT_MINMAX HELP_TCARD
HCBT_MOVESIZE HELP_TCARD
HCBT_QS HELP_TCARD
HCBT_SETFOCUS HELP_TCARD
HCBT_SETWINPOS HELP_TCARD
HCBT_SETWINPOS HELP_TCARD
HCBT_SYSCOMMAND HELP_WM_HELP
HCF_AVAILABLE HELPINFO_MENUITEM
HCF_CONFIRMHOTKEY HELPINFO_WINDOW
HCF_HIGHCONTRASTON HHT_ABOVE
HCF_HOTKEYACTIVE HHT_BELOW
HCF_HOTKEYAVAILABLE HHT_NOWHERE
HCF_HOTKEYSOUND HHT_ONDIVIDER
HCF_INDICATOR HHT_ONDIVOPEN
HDF_BITMAP HHT_ONFILTER
HDF_BITMAP_ON_RIGHT HHT_ONFILTERBUTTON
HDF_CENTER HHT_ONHEADER
HDF_IMAGE HHT_TOLEFT
HDF_JUSTIFYMASK HHT_TORIGHT
HDF_LEFT HICF_ACCELERATOR
HDF_OWNERDRAW HICF_ARROWKEYS
HDF_RIGHT HICF_DUPACCEL
HDF_RTLREADING HICF_ENTERING
HDF_SORTDOWN HICF_LEAVING
HDF_SORTUP HICF_MOUSE
HDF_STRING HICF_MOUSE
HDFT_HASNOVALUE HICF_OTHER
HDFT_ISNUMBER HICF_RESELECT
HDFT_ISSTRING HICF_TOGGLEDROPDOWN
HDI_BITMAP HIDE_WINDOW
HDI_DT_SETITEM HIMETRIC_PER_INCH
HDI_FILTER HIST_ADDTOFAVORITES
HDI_FORMAT HIST_BACK
HDI_HEIGHT HIST_FAVORITES
HDI_IMAGE HIST_FORWARD
HDI_LPARAM HIST_VIEWTREE
HDI_ORDER HKCOMB_A
HDI_TEXT HKCOMB_C
HDI_WIDTH HKCOMB_CA
HDM_CLEARFILTER HKCOMB_NONE
HDM_CREATEDRAGIMAGE HKCOMB_S
HDM_DELETEDITEM HKCOMB_SA
HDM_EDITFILTER HKCOMB_SC
HDM_FIRST HKCOMB_SCA
HDM_GETBITMAPMARGIN HKL_NEXT
HDM_GETIMAGELIST HKL_PREV
HDM_GETITEM HKM_GETHOTKEY
HDM_GETITEMA HKM_SETHOTKEY
HDM_GETITEMCOUNT HKM_SETRULES
HDM_GETITEMRECT HOLLOW_BRUSH
HDM_GETITEMW HORZRES
HDM_GETORDERARRAY HORZSIZE
<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDM_GETUNICODEFORMAT</td>
<td>HOTKEYF_ALT</td>
</tr>
<tr>
<td>HDM_HITTEST</td>
<td>HOTKEYF_CONTROL</td>
</tr>
<tr>
<td>HDM_INSERTITEM</td>
<td>HOTKEYF_EXT</td>
</tr>
<tr>
<td>HDM_INSERTITEMA</td>
<td>HOTKEYF_SHIFT</td>
</tr>
<tr>
<td>HDM_INSERTITEMW</td>
<td>HOVER_DEFAULT</td>
</tr>
<tr>
<td>HDM_LAYOUT</td>
<td>HS_BDIAGONAL</td>
</tr>
<tr>
<td>HDM_ORDERTOINDEX</td>
<td>HS_CROSS</td>
</tr>
<tr>
<td>HDM_SETBITMAPMARGIN</td>
<td>HS_DIAVCROSS</td>
</tr>
<tr>
<td>HDM_SETFILTERCHANGETIMEOUT</td>
<td>HS_FDIAGONAL</td>
</tr>
<tr>
<td>HDM_SETHOTDIVIDER</td>
<td>HS_HORIZONAL</td>
</tr>
<tr>
<td>HDM_SETIMAGELIST</td>
<td>HS_VERTICAL</td>
</tr>
<tr>
<td>HDM_SETITEM</td>
<td>HSHELL_ACCESSIBILITYSTATE</td>
</tr>
<tr>
<td>HDM_SETITEMA</td>
<td>HSHELL_ACTIVATESHELLWINDOW</td>
</tr>
<tr>
<td>HDM_SETITEMW</td>
<td>HSHELL_APPCOMMAND</td>
</tr>
<tr>
<td>HDM_SETORDERARRAY</td>
<td>HSHELL_ENDTASK</td>
</tr>
<tr>
<td>HDM_SETUNICODEFORMAT</td>
<td>HSHELL_FLASH</td>
</tr>
<tr>
<td>HDN_BEGINDRAG</td>
<td>HSHELL_GETMINRECT</td>
</tr>
<tr>
<td>HDN_BEGINTRAY</td>
<td>HSHELL_HIGHBIT</td>
</tr>
<tr>
<td>HDN_BEGINTRACKA</td>
<td>HSHELL_LANGUAGE</td>
</tr>
<tr>
<td>HDN_BEGINTRACKW</td>
<td>HSHELL_REDRAW</td>
</tr>
<tr>
<td>HDN_DIVIDERDBLCLICK</td>
<td>HSHELL_RUDEAPPACTIVATED</td>
</tr>
<tr>
<td>HDN_DIVIDERDBLCLICKA</td>
<td>HSHELL_SYSEMENUSH</td>
</tr>
<tr>
<td>HDN_DIVIDERDBLCLICKW</td>
<td>HSHELL_TASKMAN</td>
</tr>
<tr>
<td>HDN_ENDDRAG</td>
<td>HSHELL_WINDOWACTIVATED</td>
</tr>
<tr>
<td>HDN_ENDTRAY</td>
<td>HSHELL_WINDOWCREATED</td>
</tr>
<tr>
<td>HDN_ENDTRACKA</td>
<td>HSHELL_WINDOWDESTROYED</td>
</tr>
<tr>
<td>HDN_ENDTRACKW</td>
<td>HSHELL_WINDOWREPLACED</td>
</tr>
<tr>
<td>HDN_FILTERBTNCLICK</td>
<td>HSHELL_WINDOWREPLACING</td>
</tr>
<tr>
<td>HDN_FILTERCHANGE</td>
<td>HTBORDER</td>
</tr>
<tr>
<td>HDN_GETDISPINFO</td>
<td>HTBOTTOM</td>
</tr>
<tr>
<td>HDN_GETDISPINFOA</td>
<td>HTBOTTOMLEFT</td>
</tr>
<tr>
<td>HDN_GETDISPINFOW</td>
<td>HTBOTTOMRIGHT</td>
</tr>
<tr>
<td>HDN_ITEMCHANGED</td>
<td>HTCAPTION</td>
</tr>
<tr>
<td>HDN_ITEMCHANGEDA</td>
<td>HTCLIENT</td>
</tr>
<tr>
<td>HDN_ITEMCHANGEDW</td>
<td>HTCLOSE</td>
</tr>
<tr>
<td>HDN_ITEMCHANGING</td>
<td>HTERROR</td>
</tr>
<tr>
<td>HDN_ITEMCHANGINGA</td>
<td>HTGROWBOX</td>
</tr>
<tr>
<td>HDN_ITEMCHANGINGW</td>
<td>HTHELP</td>
</tr>
<tr>
<td>HDN_ITEMCLICK</td>
<td>HTHSCROLL</td>
</tr>
<tr>
<td>HDN_ITEMCLICKA</td>
<td>HTLEFT</td>
</tr>
<tr>
<td>HDN_ITEMCLICKW</td>
<td>HTMAXBUTTON</td>
</tr>
<tr>
<td>HDN_ITEMDBLCLICK</td>
<td>HTMENU</td>
</tr>
<tr>
<td>HDN_ITEMDBLCLICKA</td>
<td>HTMINBUTTON</td>
</tr>
<tr>
<td>HDN_ITEMDBLCLICKW</td>
<td>HTNOWHERE</td>
</tr>
<tr>
<td>HDN_TRACK</td>
<td>HTOBJECT</td>
</tr>
<tr>
<td>HDN_TRACKA</td>
<td>HTREDUCE</td>
</tr>
<tr>
<td>Constants</td>
<td>Values</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>HDN_TRACKW</td>
<td>HTRIGHT</td>
</tr>
<tr>
<td>HDS_BUTTONS</td>
<td>HTSIZE</td>
</tr>
<tr>
<td>HDS_DRAGDROP</td>
<td>HTSIZEFIRST</td>
</tr>
<tr>
<td>HDS_FILTERBAR</td>
<td>HTSIZELAST</td>
</tr>
<tr>
<td>HDS_FLAT</td>
<td>HTSYSMENU</td>
</tr>
<tr>
<td>HDS_FULLDRAG</td>
<td>HTTOP</td>
</tr>
<tr>
<td>HDS_HIDDEN</td>
<td>HTTOPLEFT</td>
</tr>
<tr>
<td>HDS_HORZ</td>
<td>HTTOPRIGHT</td>
</tr>
<tr>
<td>HDS_HOTTRACK</td>
<td>HTTRANSPARENT</td>
</tr>
<tr>
<td>HEBREW_CHARSET</td>
<td>HTVSCROLL</td>
</tr>
<tr>
<td>HELP_COMMAND</td>
<td>HTZOOM</td>
</tr>
<tr>
<td>I_CHILDRENCALLBACK</td>
<td>ILC_COLOR4</td>
</tr>
<tr>
<td>I_GROUPIDCALLBACK</td>
<td>ILC_COLOR8</td>
</tr>
<tr>
<td>I_GROUPIDNONE</td>
<td>ILC_COLORDDB</td>
</tr>
<tr>
<td>I_IMAGECALLBACK</td>
<td>ILC_MASK</td>
</tr>
<tr>
<td>I_IMAGENONE</td>
<td>ILC_MIRROR</td>
</tr>
<tr>
<td>I_INDENTCALLBACK</td>
<td>ILC_PALETTE</td>
</tr>
<tr>
<td>ICC_ANIMATE_CLASS</td>
<td>ILC_PERITEMMIRROR</td>
</tr>
<tr>
<td>ICC_BAR_CLASSES</td>
<td>ILCF_MOVE</td>
</tr>
<tr>
<td>ICC_COOL_CLASSES</td>
<td>ILCF_SWAP</td>
</tr>
<tr>
<td>ICC_DATE_CLASSES</td>
<td>ILD_BLEND</td>
</tr>
<tr>
<td>ICC_HOTKEY_CLASS</td>
<td>ILD_BLEND25</td>
</tr>
<tr>
<td>ICC_INTERNET_CLASSES</td>
<td>ILD_BLEND50</td>
</tr>
<tr>
<td>ICC_LINK_CLASS</td>
<td>ILD_DPISCALE</td>
</tr>
<tr>
<td>ICC_LISTVIEW_CLASSES</td>
<td>ILD_FOCUS</td>
</tr>
<tr>
<td>ICC_NATIVEFNTCTL_CLASS</td>
<td>ILD_IMAGE</td>
</tr>
<tr>
<td>ICC_PAGESCROLLER_CLASS</td>
<td>ILD_MASK</td>
</tr>
<tr>
<td>ICC_PROGRESS_CLASS</td>
<td>ILD_NORMAL</td>
</tr>
<tr>
<td>ICC_STANDARD_CLASSES</td>
<td>ILD_OVERLAYMASK</td>
</tr>
<tr>
<td>ICC_TAB_CLASSES</td>
<td>ILD_PRESERVEALPHA</td>
</tr>
<tr>
<td>ICC_TREEVIEW_CLASSES</td>
<td>ILD_ROP</td>
</tr>
<tr>
<td>ICC_UPDOWN_CLASS</td>
<td>ILD_SCALE</td>
</tr>
<tr>
<td>ICC_USEREX_CLASSES</td>
<td>ILD_SELECTED</td>
</tr>
<tr>
<td>ICC_WIN95_CLASSES</td>
<td>ILD_TRANSPARENT</td>
</tr>
<tr>
<td>ICM_ADDPROFILE</td>
<td>ILLUMINANT_A</td>
</tr>
<tr>
<td>ICM_DELETEPROFILE</td>
<td>ILLUMINANT_B</td>
</tr>
<tr>
<td>ICM_DONE_OUTSIDEDDC</td>
<td>ILLUMINANT_C</td>
</tr>
<tr>
<td>ICM_OFF</td>
<td>ILLUMINANT_D50</td>
</tr>
<tr>
<td>ICM_ON</td>
<td>ILLUMINANT_D55</td>
</tr>
<tr>
<td>ICM_QUERY</td>
<td>ILLUMINANT_D65</td>
</tr>
<tr>
<td>ICM_QUERYMATCH</td>
<td>ILLUMINANT_D75</td>
</tr>
<tr>
<td>ICM_QUERYPROFILE</td>
<td>ILLUMINANT_DAYLIGHT</td>
</tr>
<tr>
<td>ICMREGISTERICMATCHER</td>
<td>ILLUMINANT_DEVICE_DEFAULT</td>
</tr>
</tbody>
</table>
Constants

ICM_SETDEFAULTPROFILE  ILLUMINANT_F2
ICM_UNREGISTERICMATCHER  ILLUMINANT_FLUORESCENT
ICON_BIG  ILLUMINANT_MAX_INDEX
ICON_SMALL  ILLUMINANT_NTSC
ICON_SMALL2  ILLUMINANT_TUNGSTEN
IDABORT  ILS_ALPHA
IDANI_CAPTION  ILS_GLOW
IDANI_OPEN  ILS_NORMAL
IDB_HIST_LARGE_COLOR  ILS_SATURATE
IDB_HIST_SMALL_COLOR  ILS_SHADOW
IDB_STD_LARGE_COLOR  IMAGE_BITMAP
IDB_STD_SMALL_COLOR  IMAGE_CURSOR
IDB_VIEW_LARGE_COLOR  IMAGE_ENHMETAFILE
IDB_VIEW_SMALL_COLOR  IMAGE_ICON
IDCANCEL  INDEXID_CONTAINER
IDCLOSE  INDEXID_OBJECT
IDCONTINUE  INFOTIPSIZE
IDH_CANCEL  INPUT_HARDWARE
IDH_GENERIC_HELP_BUTTON  INPUT_KEYBOARD
IDH_HELP  INPUT_MOUSE
IDH_MISSING_CONTEXT  INPUTLANGCHANGE_BACKWARD
IDH_NO_HELP  INPUTLANGCHANGE_FORWARD
IDH_OK  INPUTLANGCHANGE_SYSCHARSET
IDHELP  INVALID_LINK_INDEX
IDHOT_SNAPDESKTOP  IPM_CLEARADDRESS
IDHOT_SNAPWINDOW  IPM_GETADDRESS
IDIGNORE  IPM_ISBLANK
IDNO  IPM_SETADDRESS
IDOK  IPM_SETFOCUS
IDRETRY  IPM_SETRANGE
IDTIMEOUT  IPN_FIELDCHANGED
IDTRYAGAIN  ISMEX_CALLBACK
IDYES  ISMEX_NOSEND
ILC_COLOR  ISMEX_NOTIFY
ILC_COLOR16  ISMEX_REPLIED
ILC_COLOR24  ISMEX_SEND
ILC_COLOR32  ITALIC_FONTTYPE

J

JOHAB_CHARSET

K

KEYBOARD_OVERRUN_MAKE_CODE  KF_UP
KEYEVENTF_EXTENDEDKEY  KL_NAMELENGTH
KEYEVENTF_KEYUP  KLF_ACTIVATE
<table>
<thead>
<tr>
<th>Key Event</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEYEVENTF_SCANCODE</td>
<td>KLF_NOTELLSHELL</td>
</tr>
<tr>
<td>KEYEVENTF_UNICODE</td>
<td>KLF_REORDER</td>
</tr>
<tr>
<td>KF_ALTDOWN</td>
<td>KLF_REPLACELANG</td>
</tr>
<tr>
<td>KF_DLGMODE</td>
<td>KLF_RESET</td>
</tr>
<tr>
<td>KF_EXTENDED</td>
<td>KLF_SETFORPROCESS</td>
</tr>
<tr>
<td>KF_MENUHIDE</td>
<td>KLF_SHIFTLOCK</td>
</tr>
<tr>
<td>KF_REPEAT</td>
<td>KLF_SUBSTITUTE_OK</td>
</tr>
</tbody>
</table>

**L**

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_MAX_URL_LENGTH</td>
</tr>
<tr>
<td>LAYOUT_BITMAPORIENTATIONPRESERVED</td>
</tr>
<tr>
<td>LAYOUT_BTT</td>
</tr>
<tr>
<td>LAYOUT_ORIENTATIONMASK</td>
</tr>
<tr>
<td>LAYOUT_RTL</td>
</tr>
<tr>
<td>LAYOUT_VBH</td>
</tr>
<tr>
<td>LB_ADDFILE</td>
</tr>
<tr>
<td>LB_ADDSTRING</td>
</tr>
<tr>
<td>LB_CTLCODE</td>
</tr>
<tr>
<td>LB_DELETESTRING</td>
</tr>
<tr>
<td>LB_DIR</td>
</tr>
<tr>
<td>LB_ERR</td>
</tr>
<tr>
<td>LB_ERRSPACE</td>
</tr>
<tr>
<td>LB_FINDSTRING</td>
</tr>
<tr>
<td>LB_FINDSTRINGEXACT</td>
</tr>
<tr>
<td>LB_GETANCHORINDEX</td>
</tr>
<tr>
<td>LB_GETCARETINDEX</td>
</tr>
<tr>
<td>LB_GETCOUNT</td>
</tr>
<tr>
<td>LB_GETCURSEL</td>
</tr>
<tr>
<td>LB_GETHORIZONTALEXTENT</td>
</tr>
<tr>
<td>LB_GETITEMDATA</td>
</tr>
<tr>
<td>LB_GETITEMHEIGHT</td>
</tr>
<tr>
<td>LB_GETITEMRECT</td>
</tr>
<tr>
<td>LB_GETLISTBOXINFO</td>
</tr>
<tr>
<td>LB_GETLOCALE</td>
</tr>
<tr>
<td>LB_GETSEL</td>
</tr>
<tr>
<td>LB_GETSELINDEX</td>
</tr>
<tr>
<td>LB_GETSELITEMS</td>
</tr>
<tr>
<td>LB_GETTEXT</td>
</tr>
<tr>
<td>LB_GETTEXTLEN</td>
</tr>
<tr>
<td>LB_GETTOPINDEX</td>
</tr>
<tr>
<td>LB_INITSTORAGE</td>
</tr>
<tr>
<td>LB_INSERTSTRING</td>
</tr>
<tr>
<td>LB_ITEMFROMPOINT</td>
</tr>
<tr>
<td>LB_MSGMAX</td>
</tr>
<tr>
<td>LB_MULTIPLEADDSTRING</td>
</tr>
<tr>
<td>Constant</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>LB_OKAY</td>
</tr>
<tr>
<td>LB_RESETCONTENT</td>
</tr>
<tr>
<td>LB_SELECTSTRING</td>
</tr>
<tr>
<td>LB_SELITEMRANGE</td>
</tr>
<tr>
<td>LB_SELITEMRANGEEX</td>
</tr>
<tr>
<td>LB_SETANCHORINDEX</td>
</tr>
<tr>
<td>LB_SETANCHORINDEX</td>
</tr>
<tr>
<td>LB_SETCOLUMNWIDTH</td>
</tr>
<tr>
<td>LB_SETCOUNT</td>
</tr>
<tr>
<td>LB_SETCURSEL</td>
</tr>
<tr>
<td>LB_SETHORizontALEXTENT</td>
</tr>
<tr>
<td>LB_SETITEMDATA</td>
</tr>
<tr>
<td>LB_SETITEMHEIGHT</td>
</tr>
<tr>
<td>LB_SETLOCALE</td>
</tr>
<tr>
<td>LB_SETSEL</td>
</tr>
<tr>
<td>LB_SETTABSTOPS</td>
</tr>
<tr>
<td>LB_SETTOPINDEX</td>
</tr>
<tr>
<td>LBN_DBLCLK</td>
</tr>
<tr>
<td>LBN_ERRSPACE</td>
</tr>
<tr>
<td>LBN_KILLFOCUS</td>
</tr>
<tr>
<td>LBN_SELCANCEL</td>
</tr>
<tr>
<td>LBN_SELCHANGE</td>
</tr>
<tr>
<td>LBN_SETFOCUS</td>
</tr>
<tr>
<td>LBS_COMBOBOX</td>
</tr>
<tr>
<td>LBS_DISABLENOSCROLL</td>
</tr>
<tr>
<td>LBS_EXTENDEDSEL</td>
</tr>
<tr>
<td>LBS_HASSTRINGS</td>
</tr>
<tr>
<td>LBS_MULTICOLUMN</td>
</tr>
<tr>
<td>LBS_MULTIPLESEL</td>
</tr>
<tr>
<td>LBS_NODATA</td>
</tr>
<tr>
<td>LBS_NOINTEGRALHEIGHT</td>
</tr>
<tr>
<td>LBS_NOREDRAW</td>
</tr>
<tr>
<td>LBS_NOSEL</td>
</tr>
<tr>
<td>LBS_NOTIFY</td>
</tr>
<tr>
<td>LBS_OWNERDRAWFIXED</td>
</tr>
<tr>
<td>LBS_OWNERDRAWVARIABLE</td>
</tr>
<tr>
<td>LBS_SORT</td>
</tr>
<tr>
<td>LBS_STANDARD</td>
</tr>
<tr>
<td>LBS_USETABSTOPS</td>
</tr>
<tr>
<td>LBS_WANTKEYBOARDINPUT</td>
</tr>
<tr>
<td>LC_INTERIORS</td>
</tr>
<tr>
<td>LC_MARKER</td>
</tr>
<tr>
<td>LC_NONE</td>
</tr>
<tr>
<td>LC_POLYLINE</td>
</tr>
<tr>
<td>LC_POLYMARKER</td>
</tr>
<tr>
<td>LC_STYLED</td>
</tr>
<tr>
<td>Constants</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>LC_WIDE</td>
</tr>
<tr>
<td>LC_WIDESTYLED</td>
</tr>
<tr>
<td>LF_FACESIZE</td>
</tr>
<tr>
<td>LF_FULLFACESIZE</td>
</tr>
<tr>
<td>LIF_ITEMID</td>
</tr>
<tr>
<td>LIF_ITEMINDEX</td>
</tr>
<tr>
<td>LIF_STATE</td>
</tr>
<tr>
<td>LIF_URL</td>
</tr>
<tr>
<td>LINECAPS</td>
</tr>
<tr>
<td>LIS_ENABLED</td>
</tr>
<tr>
<td>LIS_FOCUSED</td>
</tr>
<tr>
<td>LIS_VISITED</td>
</tr>
<tr>
<td>LLKHF_ALTDOWN</td>
</tr>
<tr>
<td>LLKHF_EXTENDED</td>
</tr>
<tr>
<td>LLKHF_INJECTED</td>
</tr>
<tr>
<td>LLKHF_UP</td>
</tr>
<tr>
<td>LLMHF_INJECTED</td>
</tr>
<tr>
<td>LM_GETIDEALHEIGHT</td>
</tr>
<tr>
<td>LM_GETITEM</td>
</tr>
<tr>
<td>LM_HITTEST</td>
</tr>
<tr>
<td>LM_SETITEM</td>
</tr>
<tr>
<td>LOGPIXELSX</td>
</tr>
<tr>
<td>LOGPIXELOSY</td>
</tr>
<tr>
<td>LPD_DOUBLEBUFFER</td>
</tr>
<tr>
<td>LPD_SHARE_ACCUM</td>
</tr>
<tr>
<td>LPD_SHARE_DEPTH</td>
</tr>
<tr>
<td>LPD_SHARE_STENCIL</td>
</tr>
<tr>
<td>LPD_STEREO</td>
</tr>
<tr>
<td>LPD_SUPPORT_GDI</td>
</tr>
<tr>
<td>LPD_SUPPORT_OPENGL</td>
</tr>
<tr>
<td>LPD_SWAP_COPY</td>
</tr>
<tr>
<td>LPD_SWAP_EXCHANGE</td>
</tr>
<tr>
<td>LPD_TRANSPARENT</td>
</tr>
<tr>
<td>LPD_TYPE_COLORINDEX</td>
</tr>
<tr>
<td>LPD_TYPE_RGBA</td>
</tr>
<tr>
<td>LR_COLOR</td>
</tr>
<tr>
<td>LR_COPYDELETEORG</td>
</tr>
<tr>
<td>LR_COPYFROMRESOURCE</td>
</tr>
<tr>
<td>LR_COPYRETURNORG</td>
</tr>
<tr>
<td>LR_CREATEDISSECTION</td>
</tr>
<tr>
<td>LR_DEFAULTCOLOR</td>
</tr>
<tr>
<td>LR_DEFAULTSIZE</td>
</tr>
<tr>
<td>LR_LOADFROMFILE</td>
</tr>
<tr>
<td>LR_LOADMAP3DCOLORS</td>
</tr>
<tr>
<td>LR_LOADTRANSPARENT</td>
</tr>
<tr>
<td>LR_MONOCHROME</td>
</tr>
<tr>
<td>Constant</td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td>LR_SHARED</td>
</tr>
<tr>
<td>LR_VGACOLOR</td>
</tr>
<tr>
<td>LFW_Lock</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_Workareas</td>
</tr>
<tr>
<td>LV_VIEW_DETAILS</td>
</tr>
<tr>
<td>LV_VIEW_ICON</td>
</tr>
<tr>
<td>LV_VIEW_LIST</td>
</tr>
<tr>
<td>LV_VIEW_MAX</td>
</tr>
<tr>
<td>LV_VIEW_SMALLICON</td>
</tr>
<tr>
<td>LSFW_LOCK</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LTGRAY_BRUSH</td>
</tr>
<tr>
<td>LV_MAX_WORKAREAS</td>
</tr>
<tr>
<td>LSFW_UNLOCK</td>
</tr>
<tr>
<td>LVFI_WRAP</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>LVGA_FOOTER_CENTER</td>
</tr>
<tr>
<td>LVGA_FOOTER_LEFT</td>
</tr>
<tr>
<td>LVGA_FOOTER_RIGHT</td>
</tr>
<tr>
<td>LVGA_HEADER_CENTER</td>
</tr>
<tr>
<td>LVGA_HEADER_LEFT</td>
</tr>
<tr>
<td>LVGA_HEADER_RIGHT</td>
</tr>
<tr>
<td>LVGF_ALIGN</td>
</tr>
<tr>
<td>LVGF_FOOTER</td>
</tr>
<tr>
<td>LVGF_GROUPID</td>
</tr>
<tr>
<td>LVGF_HEADER</td>
</tr>
<tr>
<td>LVGF_NONE</td>
</tr>
<tr>
<td>LVGF_STATE</td>
</tr>
<tr>
<td>LVGIT_UNFOLDED</td>
</tr>
<tr>
<td>LVGMF_BORDERCOLOR</td>
</tr>
<tr>
<td>LVGMF_BORDERSIZE</td>
</tr>
<tr>
<td>LVGMF_NONE</td>
</tr>
<tr>
<td>LVGMF_TEXTCOLOR</td>
</tr>
<tr>
<td>LVGS_COLLAPSED</td>
</tr>
<tr>
<td>LVGS_HIDDEN</td>
</tr>
<tr>
<td>LVGS_NORMAL</td>
</tr>
<tr>
<td>LVHT_ABOVE</td>
</tr>
<tr>
<td>LVHT BELOW</td>
</tr>
<tr>
<td>LVHT_NOWHERE</td>
</tr>
<tr>
<td>LVHT_ONITEM</td>
</tr>
<tr>
<td>LVHT_ONITEMICON</td>
</tr>
<tr>
<td>LVHT_ONITEMLABEL</td>
</tr>
<tr>
<td>LVHT_ONITEMSTATEICON</td>
</tr>
<tr>
<td>LVHT_TOLEFT</td>
</tr>
<tr>
<td>LVHT_TORIGHT</td>
</tr>
<tr>
<td>LVIF_COLUMNS</td>
</tr>
<tr>
<td>LVIF_DI_SETITEM</td>
</tr>
<tr>
<td>LVIF_GROUPID</td>
</tr>
<tr>
<td>LVIF_IMAGE</td>
</tr>
<tr>
<td>LVIF_INDENT</td>
</tr>
<tr>
<td>LVIF_NORECOMPUTE</td>
</tr>
<tr>
<td>LVIF_PARAM</td>
</tr>
<tr>
<td>LVIF_STATE</td>
</tr>
<tr>
<td>LVIF_TEXT</td>
</tr>
<tr>
<td>LVIM_AFTER</td>
</tr>
<tr>
<td>LVIR_BOUNDS</td>
</tr>
<tr>
<td>LVIR_ICON</td>
</tr>
<tr>
<td>LVIR_LABEL</td>
</tr>
<tr>
<td>LVIR_SELECTBOUNDS</td>
</tr>
<tr>
<td>LVIS_ACTIVATING</td>
</tr>
<tr>
<td>LVIS_CUT</td>
</tr>
<tr>
<td>Constants</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>LVIS_DROPHILITED</td>
</tr>
<tr>
<td>LVIS_FOCUSED</td>
</tr>
<tr>
<td>LVIS_GLOW</td>
</tr>
<tr>
<td>LVIS_OVERLAYMASK</td>
</tr>
<tr>
<td>LVIS_SELECTED</td>
</tr>
<tr>
<td>LVIS_STATEIMAGEMASK</td>
</tr>
<tr>
<td>LVIS_ALT</td>
</tr>
<tr>
<td>LVKF_CONTROL</td>
</tr>
<tr>
<td>LVKF_SHIFT</td>
</tr>
<tr>
<td>LVM_APPROXIMATEVIEWRECT</td>
</tr>
<tr>
<td>LVMINDOWFIELD</td>
</tr>
<tr>
<td>LVM_CANCELEDITLABEL</td>
</tr>
<tr>
<td>LVM_CREATEDRAGIMAGE</td>
</tr>
<tr>
<td>LVM_DELETEALLITEMS</td>
</tr>
<tr>
<td>LVM_DELETECOLUMN</td>
</tr>
<tr>
<td>LVM_DELETEEDITITEM</td>
</tr>
<tr>
<td>LVM_EDITLABEL</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>MA_ACTIVATE</td>
</tr>
<tr>
<td>MA_NOACTIVATE</td>
</tr>
<tr>
<td>MAC_CHARSET</td>
</tr>
<tr>
<td>MAX_LINKID_TEXT</td>
</tr>
<tr>
<td>MAX_PATH</td>
</tr>
<tr>
<td>MAXSTRETCHBLTMODE</td>
</tr>
<tr>
<td>MB_ABORTRETRYIGNORE</td>
</tr>
<tr>
<td>MB_APPLMODAL</td>
</tr>
<tr>
<td>MBCANCELTRYCONTINUE</td>
</tr>
<tr>
<td>MB_DEFAULT_DESKTOP_ONLY</td>
</tr>
<tr>
<td>MB_DEFBUTTON1</td>
</tr>
<tr>
<td>MB_DEFBUTTON2</td>
</tr>
<tr>
<td>MB_DEFBUTTON3</td>
</tr>
<tr>
<td>MB_DEFBUTTON4</td>
</tr>
<tr>
<td>MB_DEFMASK</td>
</tr>
<tr>
<td>MB_HELP</td>
</tr>
<tr>
<td>MB_ICONASTERISK</td>
</tr>
<tr>
<td>MB_ICONERROR</td>
</tr>
<tr>
<td>MB_ICONEXCLAMATION</td>
</tr>
<tr>
<td>MB_ICONHAND</td>
</tr>
<tr>
<td>MB_ICONHILITE</td>
</tr>
<tr>
<td>MB_ICONINSERT</td>
</tr>
<tr>
<td>MB_ICONQUESTION</td>
</tr>
</tbody>
</table>
MB_ICONSTOP  MF_MENUBREAK
MB_ICONWARNING MF_MOUSESELECT
MB_MISCMASK  MF.OwnerDraw
MB_MODEMASK  MF.POPUP
MB_NOFOCUS  MF.REMOVE
MB_OK  MF_RightJustify
MB_OKCANCEL  MF_SEPARATOR
MB_RIGHT  MF_SYSMENU
MB_RTLREADING  MF_UNCHECKED
MB_SERVICE_NOTIFICATION  MF.UNHILITE
MB_SERVICE_NOTIFICATION_NT3X  MF_USECHECKBITMAPS
MB_SETFOREGROUND  MFCOMMENT
MB_SYSTEMMODAL  MFS_Checked
MB_TASKMODAL  MFS_Default
MB_TOPMOST  MFS_Disabled
MB_TYPEmask  MFS_ENABLED
MB_USERICON  MFS_GRAYED
MB_YESNO  MFS_HILITE
MB_YESNOCANCEL  MFS.Unchecked
MCHT_CALENDAR  MFT_BITMAP
MCHT_CALENDARDBK  MFT_MENUBARBREAK
MCHT_CALENDARDATE  MFT_MENUBREAK
MCHT_CALENDARDATENEXT  MFT_OWNERDRAW
MCHT_CALENDARDATEPREV  MFT_RADIOCHECK
MCHT_CALENDARDAY  MFT_RIGHTCHECK
MCHT_CALENDARWEEKNUM  MFT_RIGHTJUSTIFY
MCHT_NEXT  MFT_RIGHTORDER
MCHT_NOWHERE  MFT_SEPARATOR
MCHT_PREV  MFT_STRING
MCHT_TITLE  MIIM_BITMAP
MCHT_TITLEBK  MIIM_CHECKMARKS
MCHT_TITLEBTNNEXT  MIIM_DATA
MCHT_TITLEBTNPREV  MIIM_FTYPE
MCHT_TITLEMONTH  MIIM_ID
MCHT_TITLYEAR  MIIM_STATE
MCHT_TODAYLINK  MIIM_STRING
MCM_FIRST  MIIM_SUBMENU
MCM_GETCOLOR  MIIM_TYPE
MCM_GETCURSEL  MIIM_APPLYTOSUBMENUS
MCM_GETFIRSTDAYOFWEEK  MIIM_BACKGROUND
MCM_GETMAXSELCOUNT  MIIM_HILITED
MCM_GETMAXTODAYWIDTH  MIIM_MAXHEIGHT
MCM_GETMINREQRECT  MIIM_MENU_DATA
MCM_GETMONTHDELTA  MIIM_STYLE
MCM_GETMONTHRANGE  MINSYSCOMMAND
MCM_GETRANGE MK_CONTROL
MCM_GETSELRANGE MK_LBUTTON
MCM_GETTODAY MK_MBUTTON
MCM_GETUNICODEFORMAT MK_RBUTTON
MCM_HITTEST MK_SHIFT
MCM_SETCOLOR MK_XBUTTON1
MCM_SETCURSEL MK_XBUTTON2
MCM_SETDAYSTATE MKF_AVAILABLE
MCM_SETFIRSTDAYOFWEEK MKF_CONFIRMHOTKEY
MCM_SETMAXSELCOUNT MKF_HOTKEYACTIVE
MCM_SETMONTHDELTA MKF_HOTKEYSOUND
MCM_SETRANGE MKF_INDICATOR
MCM_SETSELRANGE MKF_LEFTBUTTONDOWN
MCM_SETTODAY MKF_LEFTBUTTONSEL
MCM_SETUNICODEFORMAT MKF_MODIFIERS
MCN_GETDAYSTATE MKF_MOUSEKEYSON
MCN_SELCHANGE MKF_MOUSEMODE
MCN_SELECT MKF_REPLACENUMBERS
MCS_DAYSTATE MKF_RIGHTBUTTONDOWN
MCS_MULTISELECT MKF_RIGHTBUTTONSEL
MCS_NOTODAY MM_ANISOTROPIC
MCS_NOTODAYCIRCLE MM_HIENGLISH
MCS_WEEKNUMBERS MM_HIMETRIC
MCSC_BACKGROUND MM_ISO
MCSC_MONTHBK MM_LOENGLISH
MCSC_TEXT MM_LOMETRIC
MCSC_TITLEBK MM_MAX
MCSC_TITLETEXT MM_MAX_AXES_NAMELEN
MCSC_TRAILINGTEXT MM_MAX_FIXEDSCALE
MDIS_ALLCHILDSTYLES MM_MAX_NUMAXES
MDITILE_HORIZONTAL MM_MIN
MDITILE_SKIPDISABLED MM_TEXT
MDITILE_VERTICAL MM_TWIPS
MDITILE_ZORDER MN_GETHMENU
META_ANIMATEPALETTE MNC_CLOSE
META_ARC MNC_EXECUTE
META.BitBLT MNC_IGNORE
META.CHORD MNC_SELECT
META_CREATEBRUSHINDIRECT MND_CONTINUE
META_CREATEFONTINDIRECT MND_ENDMENU
META_CREATEPALETTE MNGOF_NOERROR
META_CREATEPATTERNBRUSH MNGO_NOINTERFACE
META_CREATEPENINDIRECT MNGOF_BOTTOMGAP
META_CREATEREGION MNGOF_TOPGAP
META_DELETEOBJECT MNS_AUTODISMISS
META_DIBBITBLT MNS_CHECKORBMP
META_DIBCREATEPATTERNBRUSH MNS_DRAGDROP
META_DIBSTRETCHBLT MNS_MODELESS
META_ELLIPSE MNS_NOCHECK
META_ESCAPE MNS_NOTIFYBYPOS
META_EXCLUDECLIPRECT MOD_ALT
META_EXTFLOODFILL MOD_CONTROL
META_EXTTEXTOUT MOD_SHIFT
META_FILLREGION MOD_WIN
META_FLOODFILL MONITOR_DEFAULTTONEAREST
META_FRAMEREGION MONITOR_DEFAULTTONULL
META_INTERSECTCLIPRECT MONITOR_DEFAULTTOPRIMARY
META_INVERTREGION MONITORINFOF_PRIMARY
META_LINETO MONO_FONT
META_MOVETO MOUSE_ATTRIBUTES_CHANGED
META_OFFSETCLIPRGN MOUSE_MOVE_ABSOLUTE
META_OFFSETVIEWPORTORG MOUSE_MOVE_RELATIVE
META_OFFSETWINDOWORG MOUSE_VIRTUAL_DESKTOP
META_PAINTREGION MOUSEEVENTF_ABSOLUTE
META_PAINTBLT MOUSEEVENTF_LEFTDOWN
META_PIE MOUSEEVENTF_LEFTUP
META_POLYGON MOUSEEVENTF_MIDDLEDOWN
META_POLYLINE MOUSEEVENTF_MIDDLEUP
META_POLYPOLYGON MOUSEEVENTF_MOVE
META_REALIZEPALETTE MOUSEEVENTF_RIGHTDOWN
META_RECTANGLE MOUSEEVENTF_RIGHTUP
META_RESIZEPALETTE MOUSEEVENTF_VIRTUALDESKTOP
META_RESTOREDC MOUSEEVENTF_WHEEL
META_ROUNDRECT MOUSEEVENTF_XDOWN
META_SAVEDC MOUSEEVENTF_XUP
META_SCALEVIEWPORTTEXT MOUSETAILS
META_SCALEWINDOWTEXT MSGF_COMMCTRL_BEGINDRAG
META_SELECTCLIPREGION MSGF_COMMCTRL_DRAGSELECT
META_SELECTOBJECT MSGF_COMMCTRL_SIZEHEADER
META_SELECTPALETTE MSGF_COMMCTRL_TOOLBARCHAR
META_SETBKCOLOR MSGF_DIALOGBOX
META_SETBKMODE MSGF_MAX
META_SETDIBTODEV MSGF_MENU
META_SETLAYOUT MSGF_MESSAGEBOX
META_SETMAPMODE MSGF_NEXTWINDOW
META_SETMAPPERFLAGS MSGF_SCROLLBAR
META_SETPALENTRIES MSGF_USER
META_SETPIXEL MWMO_ALERTABLE
META_SETPOLYPATMODE MWMO_INPUTAVAILABLE
META_SETRELABS MWMO_WAITALL
META_SETROP2 MWT_IDENTITY
META_SETSTRETCHBLTMODE MWT_LEFTMULTIPLY
META_SETTEXTALIGN  MWT_MAX
META_SETTEXTCHAREXTRA  MWT_MIN
META_SETTEXTCOLOR  MWT_RIGHTMULTIPLY
META_SETTEXTJUSTIFICATION

N
NEWFRAME  NM_RDOWN
NEXTBAND  NM_RELEASEDCAPTURE
NF_QUERY  NM_RETURN
NF_REQUERY  NM_SETCURSOR
NFR_ANSI  NM_SETFOCUS
NF_UNICODE  NM_THEMECHANGED
NFS_ALL  NM_TOOLTIPSCREATED
NFS_BUTTON  NONANTIALIASED_QUALITY
NFS_EDIT  NTM_DSIG
NFS_LISTCOMBO  NTM_MULTIPLEMASTER
NFS_STATIC  NTM_NONNEGATIVE_AC
NFS_USEFONTASSOC  NTM_PS_OPENTYPE
NM_CHAR  NTM_TT_OPENTYPE
NM_CLICK  NTM_TYPE1
NM_CUSTOMDRAW  NULL
NM_DBLCLK  NULL_BRUSH
NM_HOVER  NULL_PEN
NM_KEYDOWN  NULLREGION
NM_KILLFOCUS  NUMBRUSHES
NM_LDOWN  NUMCOLORS
NM_NCHITTEST  NUMFONTS
NM_OUTOFMEMORY  NUMMARKERS
NM_RCLICK  NUMPENS
NM_RDBLCLK  NUMRESERVED

O
OBJ_BITMAP  ODS_FOCUS
OBJ_BRUSH  ODS_GRAYED
OBJ_COLORSPACE  ODS_HOTLIGHT
OBJ_DC  ODS_INACTIVE
OBJ_ENHMETADC  ODS_NOACCEL
OBJ_ENHMETAFILE  ODS_NOFOCUSRECT
OBJ_EXTPEX  ODS_SELECTED
OBJ_FONT  ODT_BUTTON
OBJ_MEMDC  ODT_COMBOBOX
OBJ_METADC  ODT_HEADER
OBJ_METAFILE  ODT_LISTBOX
OBJ_PAL  ODT_LISTVIEW
OBJ_PEN  ODT_MENU
<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ_REGION</td>
<td>ODT_STATIC</td>
</tr>
<tr>
<td>OBM_BTNCORNERS</td>
<td>ODT_TAB</td>
</tr>
<tr>
<td>OBM_BTSIZE</td>
<td>OEM_CHARSET</td>
</tr>
<tr>
<td>OBM_CHECK</td>
<td>OEM_FIXED_FONT</td>
</tr>
<tr>
<td>OBM_CHECKBOXES</td>
<td>OFN_ALLOWMULTISELECT</td>
</tr>
<tr>
<td>OBM_CLOSE</td>
<td>OFN_CREATEPROMPT</td>
</tr>
<tr>
<td>OBM_COMBO</td>
<td>OFN_DONTADDTORECENT</td>
</tr>
<tr>
<td>OBM_DNARROW</td>
<td>OFN_ENABLEHOOK</td>
</tr>
<tr>
<td>OBM_DNARROWD</td>
<td>OFN_ENABLEINCLUDENOTIFY</td>
</tr>
<tr>
<td>OBM_DNARROWI</td>
<td>OFN_ENABLETEMPLATE</td>
</tr>
<tr>
<td>OBM_LFARROW</td>
<td>OFN_ENABLETEMPLATEHANDLE</td>
</tr>
<tr>
<td>OBM_LFARROWD</td>
<td>OFN_EX_NOPROGRESSBAR</td>
</tr>
<tr>
<td>OBM_MNARROW</td>
<td>OFN_EXPLORER</td>
</tr>
<tr>
<td>OBM_OLD_CLOSE</td>
<td>OFN_EXTENSIONDIFFERENT</td>
</tr>
<tr>
<td>OBM_OLD_DNARROW</td>
<td>OFN_FILEMUSTEXIST</td>
</tr>
<tr>
<td>OBM_OLD_LFARROW</td>
<td>OFN_FORCEHIDEREADONLY</td>
</tr>
<tr>
<td>OBM_OLD_REDUCE</td>
<td>OFN_HIDEERROR</td>
</tr>
<tr>
<td>OBM_OLD_RESTORE</td>
<td>OFN_LONGNAMES</td>
</tr>
<tr>
<td>OBM_OLD_RGARROW</td>
<td>OFN_NOCHANGEDIR</td>
</tr>
<tr>
<td>OBM_OLD_UPARROW</td>
<td>OFN_NODIRECTLINKS</td>
</tr>
<tr>
<td>OBM_OLD_ZOOM</td>
<td>OFN_NOLONGLINES</td>
</tr>
<tr>
<td>OBM_REDUCE</td>
<td>OFN_NONETWORKBUTTON</td>
</tr>
<tr>
<td>OBM_REDECREASE</td>
<td>OFN_NOREADONLYRETURN</td>
</tr>
<tr>
<td>OBM_RESTORE</td>
<td>OFN_NOTESTFILECREATE</td>
</tr>
<tr>
<td>OBM_RESTORED</td>
<td>OFN_NOVALIDATE</td>
</tr>
<tr>
<td>OBM_RGARROW</td>
<td>OFN_OVERWRITEPROMPT</td>
</tr>
<tr>
<td>OBM_RGARROWD</td>
<td>OFN_PATHMUSTEXIST</td>
</tr>
<tr>
<td>OBM_RGARROWI</td>
<td>OFN_READONLY</td>
</tr>
<tr>
<td>OBM_SIZE</td>
<td>OFN_SHAREAWARE</td>
</tr>
<tr>
<td>OBM_UPARROW</td>
<td>OFN_SHAREFALLTHROUGH</td>
</tr>
<tr>
<td>OBM_UPARROWD</td>
<td>OFN_SHARENOWARN</td>
</tr>
<tr>
<td>OBM_UPARROWI</td>
<td>OFN_SHAREWARN</td>
</tr>
<tr>
<td>OBM_ZOOM</td>
<td>OFN_SHOWHELP</td>
</tr>
<tr>
<td>OBM_ZOOMD</td>
<td>OIC_BANG</td>
</tr>
<tr>
<td>OCR_APPSTARTING</td>
<td>OIC_BANG</td>
</tr>
<tr>
<td>OCR_CROSS</td>
<td>OIC_ERROR</td>
</tr>
<tr>
<td>OCR_HAND</td>
<td>OIC_HAND</td>
</tr>
<tr>
<td>OCR_IBEAM</td>
<td>OIC_INFORMATION</td>
</tr>
<tr>
<td>OCR_ICOCUR</td>
<td>OIC_NOTE</td>
</tr>
<tr>
<td>OCR_ICON</td>
<td>OIC_QUES</td>
</tr>
<tr>
<td>OCR_IC.ERROR</td>
<td>OIC_SAMPLE</td>
</tr>
<tr>
<td>OCR_NORMAL</td>
<td>OIC_WARNING</td>
</tr>
<tr>
<td>OCR_SIZE</td>
<td>OIC_WINLOGO</td>
</tr>
<tr>
<td>OCR_SIZE;/ALL</td>
<td>OPAQUE</td>
</tr>
<tr>
<td>OCR_SIZE;/ALL</td>
<td>OPAQUE</td>
</tr>
<tr>
<td>OCR_SIZE;/ALL</td>
<td>OPAQUE</td>
</tr>
<tr>
<td>OCR_SIZE;/ESW</td>
<td>ORD_LANGDRIVER</td>
</tr>
<tr>
<td>Constants</td>
<td>Precis Type</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>OCR_SIZENS</td>
<td>OUT_CHARACTER_PRECIS</td>
</tr>
<tr>
<td>OCR_SIZENWSE</td>
<td>OUT_DEFAULT_PRECIS</td>
</tr>
<tr>
<td>OCR_SIZEWE</td>
<td>OUT_DEVICE_PRECIS</td>
</tr>
<tr>
<td>OCR_UP</td>
<td>OUT_OUTLINE_PRECIS</td>
</tr>
<tr>
<td>OCR_WAIT</td>
<td>OUT_PS_ONLY_PRECIS</td>
</tr>
<tr>
<td>ODA_DRAWENTIRE</td>
<td>OUT_RASTER_PRECIS</td>
</tr>
<tr>
<td>ODA_FOCUS</td>
<td>OUT_SCREEN_OUTLINE_PRECIS</td>
</tr>
<tr>
<td>ODA_SELECT</td>
<td>OUT_STRING_PRECIS</td>
</tr>
<tr>
<td>ODS_CHECKED</td>
<td>OUT_STROKE_PRECIS</td>
</tr>
<tr>
<td>ODS_COMBOBOXEDIT</td>
<td>OUT_TT_ONLY_PRECIS</td>
</tr>
<tr>
<td>ODS_DEFAULT</td>
<td>OUT_TT_PRECIS</td>
</tr>
<tr>
<td>ODS_DISABLED</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN_ANY</td>
<td>PD_USEDEVMODECOPIES</td>
</tr>
<tr>
<td>PAN_ARMSTYLE_INDEX</td>
<td>PD_USEDEVMODECOPIESANDCOLLATE</td>
</tr>
<tr>
<td>PAN_BENT_ARMS_DOUBLE_SERIF</td>
<td>PD_USELARGETEMPLATE</td>
</tr>
<tr>
<td>PAN_BENT_ARMS_HORZ</td>
<td>PDEVICEDIESE</td>
</tr>
<tr>
<td>PAN_BENT_ARMS_SINGLE_SERIF</td>
<td>PFD_DEPTH_DONTCARE</td>
</tr>
<tr>
<td>PAN_BENT_ARMS_VERT</td>
<td>PFD_DOUBLEBUFFER</td>
</tr>
<tr>
<td>PAN_BENT_ARMS_WEDGE</td>
<td>PFD_DOUBLEBUFFER_DONTCARE</td>
</tr>
<tr>
<td>PAN_CONTRAST_HIGH</td>
<td>PFD_DRAW_TO_BITMAP</td>
</tr>
<tr>
<td>PAN_CONTRAST_INDEX</td>
<td>PFD_DRAW_TO_WINDOW</td>
</tr>
<tr>
<td>PAN_CONTRAST_LOW</td>
<td>PFD_GENERIC_ACCELERATED</td>
</tr>
<tr>
<td>PAN_CONTRAST_MEDIUM</td>
<td>PFD_GENERIC_FORMAT</td>
</tr>
<tr>
<td>PAN_CONTRAST_MEDIUM_HIGH</td>
<td>PFD_MAIN_PLANE</td>
</tr>
<tr>
<td>PAN_CONTRAST_MEDIUM_LOW</td>
<td>PFD_NEED_PALETTE</td>
</tr>
<tr>
<td>PAN_CONTRAST_NONE</td>
<td>PFD_NEED_SYSTEM_PALETTE</td>
</tr>
<tr>
<td>PAN_CONTRAST_VERY_HIGH</td>
<td>PFD_OVERLAY_PLANE</td>
</tr>
<tr>
<td>PAN_CONTRAST_VERY_LOW</td>
<td>PFD_STEREO</td>
</tr>
<tr>
<td>PAN_CULTURE_LATIN</td>
<td>PFD_STEREO_DONTCARE</td>
</tr>
<tr>
<td>PAN_FAMILY_DECORATIVE</td>
<td>PFD_SUPPORT_DIRECTDRAW</td>
</tr>
<tr>
<td>PAN_FAMILY_PICTORIAL</td>
<td>PFD_SUPPORT_GDI</td>
</tr>
<tr>
<td>PAN_FAMILY_SCRIPT</td>
<td>PFD_SUPPORT_OPENGL</td>
</tr>
<tr>
<td>PAN_FAMILY_TEXT_DISPLAY</td>
<td>PFD_SWAP_COPY</td>
</tr>
<tr>
<td>PAN_FAMILYTYPE_INDEX</td>
<td>PFD_SWAP_EXCHANGE</td>
</tr>
<tr>
<td>PAN_LETT_NORMAL_BOXED</td>
<td>PFD_SWAP_LAYER_BUFFERS</td>
</tr>
<tr>
<td>PAN_LETT_NORMAL_CONTACT</td>
<td>PFD_TYPE_COLORINDEX</td>
</tr>
<tr>
<td>PAN_LETT_NORMAL_FLATTENED</td>
<td>PFD_TYPE_RGBA</td>
</tr>
<tr>
<td>PAN_LETT_NORMAL_OFF_CENTER</td>
<td>PFD_UNDERLAY_PLANE</td>
</tr>
<tr>
<td>PAN_LETT_NORMAL_ROUNDED</td>
<td>PGB_BOTTOMORRIGHT</td>
</tr>
<tr>
<td>PAN_LETT_NORMAL_SQUARE</td>
<td>PGB_TOPORLEFT</td>
</tr>
<tr>
<td>PAN_LETT_NORMAL_WEIGHTED</td>
<td>PGF_CALCHEIGHT</td>
</tr>
<tr>
<td>PAN_LETT_OBLIQUE_BOXED</td>
<td>PGF_CALCWIDTH</td>
</tr>
<tr>
<td>PAN_LETT_OBLIQUE_CONTACT</td>
<td>PGF_DEPRESSED</td>
</tr>
</tbody>
</table>
PAN_LETT_OBLIQUE_FLATTENED: PGF_GRAYED
PAN_LETT_OBLIQUE_OFF_CENTER: PGF_HOT
PAN_LETT_OBLIQUE_ROUNDED: PGF_INVISIBLE
PAN_LETT_OBLIQUE_SQUARE: PGF_NORMAL
PAN_LETTERFORM_INDEX: PGF_SCROLLDOWN
PAN_MIDLINE_CONSTANT_POINTED: PGF_SCROLLLEFT
PAN_MIDLINE_CONSTANT_SERIFED: PGF_SCROLLUP
PAN_MIDLINE_CONSTANT_TRIMMED: PGK_CONTROL
PAN_MIDLINE_HIGH_POINTED: PGK_MENU
PAN_MIDLINE_HIGH_SERIFED: PGK_SHIFT
PAN_MIDLINE_HIGH_TRIMMED: PGM_FIRST
PAN_MIDLINE_INDEX: PGM_FORWARDMOUSE
PAN_MIDLINE_LOW_POINTED: PGM_GETBKCOLOR
PAN_MIDLINE_LOW_SERIFED: PGM_GETBORDER
PAN_MIDLINE_LOW_TRIMMED: PGM_GETBUTTONSIZE
PAN_MIDLINE_STANDARD_POINTED: PGM_GETBUTTONSTATE
PAN_MIDLINE_STANDARD_SERIFED: PGM_GETDROPTARGET
PAN_MIDLINE_STANDARD_TRIMMED: PGM_GETPOS
PAN_NO_FIT: PGM_RECALCSIZE
PAN_PROP_CONDENSED: PGM_SETBKCOLOR
PAN_PROP_EVEN_WIDTH: PGM_SETBORDER
PAN_PROP_EXPANDED: PGM_SETBUTTONSIZE
PAN_PROP_MODERN: PGM_SETCHILD
PAN_PROP_MONOSPACED: PGM_SETPOS
PAN_PROP_OLD_STYLE: PGN_CALCSIZE
PAN_PROP_VERY_CONDENSED: PGN_HOTITEMCHANGE
PAN_PROP_VERY_EXPANDED: PGN_SCROLL
PAN_PROPORTION_INDEX: PGS_AUTOSCROLL
PAN_SERIF_BONE: PGS_DRAGNDROP
PAN_SERIF_COVE: PGS_HORZ
PAN_SERIF_EXAGGERATED: PGS_VERT
PAN_SERIF_FLARED: PHYSICALHEIGHT
PAN_SERIF_NORMAL_SANS: PHYSICALOFFSETX
PAN_SERIF_OBTUSE_COVE: PHYSICALOFFSETY
PAN_SERIF_OBTUSE_SANS: PHYSICALWIDTH
PAN_SERIF_OBTUSE_SQUARE_COVE: PLANES
PAN_SERIF_PERP_SANS: PM_NOREMOVE
PAN_SERIF_ROUNDED: PM_NOYIELD
PAN_SERIF_SQUARE: PM_REMOVE
PAN_SERIF_SQUARE_COVE: PMB_ACTIVE
PAN_SERIF_THIN: POLYFILL_LAST
PAN_SERIF_TRIANGLE: POLYGONALCAPS
PAN_SERIFSTYLE_INDEX: POSTSCRIPT_DATA
PAN_STRAIGHT_ARMS_DOUBLE_SERIF: POSTSCRIPT_IDENTIFY
PAN_STRAIGHT_ARMS_HORZ: POSTSCRIPT_IGNORE
<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN_STRAIGHT_ARMS_SINGLE_SERIF</td>
<td>POSTSCRIPT_INJECTION</td>
</tr>
<tr>
<td>PAN_STRAIGHT_ARMS_VERT</td>
<td>POSTSCRIPT_PASSTHROUGH</td>
</tr>
<tr>
<td>PAN_STRAIGHT_ARMS_WEDGE</td>
<td>PR_JOBSTATUS</td>
</tr>
<tr>
<td>PANROKEGRADUAL_DIAG</td>
<td>PRF_CHECKVISIBLE</td>
</tr>
<tr>
<td>PANROKEGRADUAL_HORZ</td>
<td>PRF_CHILDREN</td>
</tr>
<tr>
<td>PANROKEGRADUAL_TRAN</td>
<td>PRF_CLIENT</td>
</tr>
<tr>
<td>PANROKEGRADUAL_VERT</td>
<td>PRF_ERASEBKGND</td>
</tr>
<tr>
<td>PANROKEINSTANT_VERT</td>
<td>PRF_NONCLIENT</td>
</tr>
<tr>
<td>PANROKERAPID_HORZ</td>
<td>PRF Owned</td>
</tr>
<tr>
<td>PANROKERAPID_VERT</td>
<td>PRINTER_FONTTYPE</td>
</tr>
<tr>
<td>PANROKEVARIATION_INDEX</td>
<td>PRINTRATEUNIT_CPS</td>
</tr>
<tr>
<td>PAN_WEIGHT_BLACK</td>
<td>PRINTRATEUNIT_IPM</td>
</tr>
<tr>
<td>PAN_WEIGHT_BOLD</td>
<td>PRINTRATEUNIT_LPM</td>
</tr>
<tr>
<td>PAN_WEIGHT_BOOK</td>
<td>PRINTRATEUNIT_PPM</td>
</tr>
<tr>
<td>PAN_WEIGHT_DEMI</td>
<td>PROOF_QUALITY</td>
</tr>
<tr>
<td>PAN_WEIGHT_HEAVY</td>
<td>PS_ALTERNATE</td>
</tr>
<tr>
<td>PAN_WEIGHT_INDEX</td>
<td>PS_COSMETIC</td>
</tr>
<tr>
<td>PAN_WEIGHT_LIGHT</td>
<td>PS_DASH</td>
</tr>
<tr>
<td>PAN_WEIGHT_MEDIUM</td>
<td>PS_DASHDOT</td>
</tr>
<tr>
<td>PAN_WEIGHT_NORD</td>
<td>PS_DASHDOTDOT</td>
</tr>
<tr>
<td>PAN_WEIGHT_THIN</td>
<td>PS_DOT</td>
</tr>
<tr>
<td>PAN_WEIGHTVERYLIGHT</td>
<td>PS_ENDCAP_FLAT</td>
</tr>
<tr>
<td>PAN_XHEIGHTCONSTANT_LARGE</td>
<td>PS_ENDCAP_MASK</td>
</tr>
<tr>
<td>PAN_XHEIGHTCONSTANT_SMALL</td>
<td>PS_ENDCAP_ROUND</td>
</tr>
<tr>
<td>PAN_XHEIGHTCONSTANT_STD</td>
<td>PS_ENDCAP_SQUARE</td>
</tr>
<tr>
<td>PAN_XHEIGHT_DUCKING_LARGE</td>
<td>PS_GEOMETRIC</td>
</tr>
<tr>
<td>PAN_XHEIGHT_DUCKING_SMALL</td>
<td>PS_INSIDEFRAME</td>
</tr>
<tr>
<td>PAN_XHEIGHT_DUCKING_STD</td>
<td>PS_JOIN_BEVEL</td>
</tr>
<tr>
<td>PAN_XHEIGHT_INDEX</td>
<td>PS_JOIN_MASK</td>
</tr>
<tr>
<td>PAN_XHEIGHT_INDEX</td>
<td>PS_JOIN_MITER</td>
</tr>
<tr>
<td>PASSTHROUGH</td>
<td>PS_JOIN_ROUND</td>
</tr>
<tr>
<td>PBM_DELTAPOS</td>
<td>PS_NULL</td>
</tr>
<tr>
<td>PBM_GETPOS</td>
<td>PS_SOLID</td>
</tr>
<tr>
<td>PBM_GETRANGE</td>
<td>PS_STYLE_MASK</td>
</tr>
<tr>
<td>PBM_SETBARCOLOR</td>
<td>PS_TYPE_MASK</td>
</tr>
<tr>
<td>PBM_SETBKCOLOR</td>
<td>PS_USERSTYLE</td>
</tr>
<tr>
<td>PBM_SETMARQUEE</td>
<td>PSD_DEFAULTMINMARGINS</td>
</tr>
<tr>
<td>PBM_SETPOS</td>
<td>PSD_DISABLEMARGINS</td>
</tr>
<tr>
<td>PBM_SETRANGE</td>
<td>PSD_DISABLEORIENTATION</td>
</tr>
<tr>
<td>PBM_SETRANGE32</td>
<td>PSD_DISABLEPAGEPAINTING</td>
</tr>
<tr>
<td>PBM_SETSTEP</td>
<td>PSD_DISABLEPAPER</td>
</tr>
<tr>
<td>PBM_STEPIT</td>
<td>PSD_DISABLEPRINTER</td>
</tr>
<tr>
<td>PBS_MARQUEE</td>
<td>PSD_ENABLEPAGEPAINTHOOK</td>
</tr>
<tr>
<td>PBS_SMOOTH</td>
<td>PSD_ENABLEPAGESETUPHOOK</td>
</tr>
<tr>
<td>PBS_VERTICAL</td>
<td>PSD_ENABLEPAGESETUPTEMPLATE</td>
</tr>
<tr>
<td>PBT_APMBATTERYLOW</td>
<td>PSD_ENABLEPAGESETUPTEMPLATEHANDLE</td>
</tr>
<tr>
<td>Constant</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>PBT_APMOEVENT</td>
<td>PSD_INHUNDREDTHSOFMILLIMETERS</td>
</tr>
<tr>
<td>PBT_APMPOWERSTATUSCHANGE</td>
<td>PSD_INTHOUSANDTHSOFINCHES</td>
</tr>
<tr>
<td>PBT_APMQUERYSTANDBY</td>
<td>PSD_INWINININTLMEASURE</td>
</tr>
<tr>
<td>PBT_APMQUERYSTANDBYFAILED</td>
<td>PSD_MARGINS</td>
</tr>
<tr>
<td>PBT_APMQUERYSUSPEND</td>
<td>PSD_MINMARGINS</td>
</tr>
<tr>
<td>PBT_APMQUERYSUSPENDFAILED</td>
<td>PSD_NONETWORKBUTTON</td>
</tr>
<tr>
<td>PBT_APMRESUMEAUTOMATIC</td>
<td>PSD_NOWARNING</td>
</tr>
<tr>
<td>PBT_APMRESUMECRITICAL</td>
<td>PSD_RETURNDEFAULT</td>
</tr>
<tr>
<td>PBT_APMRESUMESTANDBY</td>
<td>PSD_SHOWHELP</td>
</tr>
<tr>
<td>PBT_APMRESUMESUSPEND</td>
<td>PSIDENT_GDICENTRIC</td>
</tr>
<tr>
<td>PBT_APMSTANDBY</td>
<td>PSIDENT_PSCENTRIC</td>
</tr>
<tr>
<td>PBT_APMSuspend</td>
<td>PSINJECT_BEGINDEFAULTS</td>
</tr>
<tr>
<td>PBT_APMRESUMEFROMFAILURE</td>
<td>PSINJECT_BEGINPAGESETUP</td>
</tr>
<tr>
<td>PC_EXPLICIT</td>
<td>PSINJECT_BEGINPROLOG</td>
</tr>
<tr>
<td>PC_INTERIORS</td>
<td>PSINJECT_BEGINSETUP</td>
</tr>
<tr>
<td>PC_NOCOLLAPSE</td>
<td>PSINJECT_BEGINSTREAM</td>
</tr>
<tr>
<td>PC_NONE</td>
<td>PSINJECT_BOUNDINGBOX</td>
</tr>
<tr>
<td>PC_PATHS</td>
<td>PSINJECT_COMMENTS</td>
</tr>
<tr>
<td>PC_POLYGON</td>
<td>PSINJECT_DOCNEEDEDRES</td>
</tr>
<tr>
<td>PC_POLYPOLYGON</td>
<td>PSINJECT_DOCSUPPLIEDRES</td>
</tr>
<tr>
<td>PC_RECTANGLE</td>
<td>PSINJECT_DOCUMENTPROCESSCOLORS</td>
</tr>
<tr>
<td>PC_RESERVED</td>
<td>PSINJECT_DOCUMENTPROCESSCOLORSATEND</td>
</tr>
<tr>
<td>PC_SCANLINE</td>
<td>PSINJECT_ENDDEFAULTS</td>
</tr>
<tr>
<td>PC_STYLED</td>
<td>PSINJECT_ENDPAGECOMMENTS</td>
</tr>
<tr>
<td>PC_TRAPEZOID</td>
<td>PSINJECT_ENDPAGESETUP</td>
</tr>
<tr>
<td>PC_WIDE</td>
<td>PSINJECT_ENDPROLOG</td>
</tr>
<tr>
<td>PC_WIDESTYLED</td>
<td>PSINJECT_ENDPAGE</td>
</tr>
<tr>
<td>PC_WINDPOLYGON</td>
<td>PSINJECT_ENDSTREAM</td>
</tr>
<tr>
<td>PD_ALLPAGES</td>
<td>PSINJECT_EOF</td>
</tr>
<tr>
<td>PD_COLLATE</td>
<td>PSINJECT_ORIENTATION</td>
</tr>
<tr>
<td>PD_CURRENTPAGE</td>
<td>PSINJECT_PAGENUMBER</td>
</tr>
<tr>
<td>PD_DISABLEPRINTTOFILE</td>
<td>PSINJECT_PAGENUMBER</td>
</tr>
<tr>
<td>PD_ENABLEPRINTHOOK</td>
<td>PSINJECT_PAGEORDER</td>
</tr>
<tr>
<td>PD_ENABLEPRINTTEMPLATE</td>
<td>PSINJECT_PAGES</td>
</tr>
<tr>
<td>PD_ENABLEPRINTTEMPLATEHANDLE</td>
<td>PSINJECT_PAGESATEND</td>
</tr>
<tr>
<td>PD_ENABLESETUPHOOK</td>
<td>PSINJECT_PAGETRAILER</td>
</tr>
<tr>
<td>PD_ENABLESETUPTEMPLATE</td>
<td>PSINJECT_PLATECOLOR</td>
</tr>
<tr>
<td>PD_ENABLESETUPTEMPLATEHANDLE</td>
<td>PSINJECT_PSDADOBE</td>
</tr>
<tr>
<td>PD_EXCLUSIONFLAGS</td>
<td>PSINJECT_SHOWPAGE</td>
</tr>
<tr>
<td>PD_HIDEPRINTTOFILE</td>
<td>PSINJECT_TRAILER</td>
</tr>
<tr>
<td>PD_NOCURRENTPAGE</td>
<td>PSINJECT_VMRSTORE</td>
</tr>
<tr>
<td>PD_NONETWORKBUTTON</td>
<td>PSINJECT_VMSAVE</td>
</tr>
<tr>
<td>PD_NOPAGENUMS</td>
<td>PSPROTOTOCOL_ASCII</td>
</tr>
<tr>
<td>PD_NOSELECTION</td>
<td>PSPROTOTOCOL_BCP</td>
</tr>
<tr>
<td>PD_NOWARNING</td>
<td>PSPROTOTOCOL_BINARY</td>
</tr>
<tr>
<td>PD_PAGENUMS</td>
<td>PSPROTOTOCOL_TBCP</td>
</tr>
</tbody>
</table>
PD_PRINTSETUP   PT_BEZIERTO
PD_PRINTTOFILE  PT_CLOSEFIGURE
PD_RESULT_APPLY PT_LINETO
PD_RESULT_CANCEL PT_MOVE
PD_RESULT_PRINT  PW_CLIENTONLY
PD_RETURNDC     PWR_CRITICALRESUME
PD_RETURNDEFAULT PWR_FAIL
PD_RETURNIC      PWR_OK
PD_SELECTION     PWR_SUSPENDREQUEST
PD_SHOWHELP      PWR_SUSPENDRESUME

Q
QDI_DIBTOSCREEN  QS_MOUSE
QDI_GETDIBITS    QS_MOUSEBUTTON
QDI_SETDIBITS   QS_MOUSEMOVE
QDI_STRETCHDIB  QS_PAINT
QS_ALLEVENTS    QS_POSTMESSAGE
QS_ALLINPUT     QS_RAWINPUT
QS_ALLPOSTMESSAGE QS_SENDMESSAGE
QS_HOTKEY       QS_TIMER
QS_INPUT        QUERYDIBSUPPORT
QS_KEY          QUERYESCSUPPORT

R
R2_BLACK         RBN_ENDDRAG
R2_COPYPEN      RBN_GETOBJECT
R2_LAST          RBN_HEIGHTCHANGE
R2_MASKNOTPEN   RBN_LAYOUTCHANGED
R2_MASKPEN      RBN_MINMAX
R2_MASKPENNOT   RBNM_ID
R2_MERGENOTPEN  RBNM_LPARAM
R2_MERGEPEN     RBNM_STYLE
R2_MERGEPENNOT  RBS_AUTOSIZE
R2_NOP          RBS_BANDBORDERS
R2_NOT          RBS_DBLCLKTOGGLE
R2_NOTCOPYPEN  RBS_FIXEDORDER
R2_NOTMASKPEN  RBS_REGISTERDROP
R2_NOTMERGEPEN RBS_TOOLTIPS
R2_NOTXORPEN   RBS_VARHEIGHT
R2_WHITE        RBS_VERTICALGRIPPER
R2_XORPEN      RBSTR_CHANGERECT
RASTER_FONTTYPE  RC_BANDING
RASTERCAPS     RC_BIGFONT
RB_BEGINDRAG   RC_BITBLT
RB_DELETEBAND  RC_BITMAP64
RB_DRAGMOVE RC_DEVBIT
RB_ENDDRAUG RC_DI_BITMAP
RB_GETBANDBORDERS RC_DIBTODEV
RB_GETBANDCOUNT RC_FLOODFILL
RB_GETBANDINFO RC_GDI20_OUTPUT
RB_GETBANDINFOA RC_GDI20_STATE
RB_GETBANDINFOW RC_OPA_DX_OUTPUT
RB_GETBANDMARGINS RC_PALETTE
RB_GETBARHEIGHT RC_SAVEBITMAP
RB_GETBARINFO RC_SCALING
RB_GETBKCOLOR RC_STRETCHBLT
RB_GETCOLORSCHEME RC_STRETCHDIB
RB_GETDROPTARGET RDH_RECTANGLES
RB_GETPALETTE RDW_ALLCHILDREN
RB_GETRECT RDW_ERASE
RB_GETROWCOUNT RDW_ERASENOW
RB_GETROWHEIGHT RDW_FRAME
RB_GETTEXTCOLOR RDW_INTERNALPAINT
RB_GETTEXTCOLORS RDW_INVALIDATE
RB_GETUNICODEFORMAT RDW_NOCHILDREN
RB_HITTEST RDW_NOERASE
RB_IDTOINDEX RDW_NOFRAME
RB_INSERTBAND RDW_NOINTERNALPAINT
RB_INSERTBANDA RDW_UPDATEDNOW
RB_INSERTBANDW RDW_VALIDATE
RB_MAXIMIZEBAND REGULAR_FONTTYPE
RB_MINIMIZEBAND RELATIVE
RB_MOVEBAND RES_CURSOR
RB_PUSHCHEVRON RES_ICON
RB_SETBANDINFO RESTORE_CTM
RB_SETBANDINFOA RGN_AND
RB_SETBANDINFOW RGN_COPY
RB_SETBARINFO RGN_DIFF
RB_SETBKCOLOR RGN_ERROR
RB_SETCOLORSCHEME RGN_MAX
RB_SETPALETTE RGN_MIN
RB_SETPARENT RGN_OR
RB_SETTEXTCOLOR RGN_XOR
RB_SETTOOLTIPS RI_KEY_BREAK
RB_SETUNICODEFORMAT RI_KEY_E0
RB_SETWINDOWTHEME RI_KEY_E1
RB_SHOWBAND RI_KEY_MAKE
RB_SIZETORECT RI_KEY_TERMSRV_SET_LED
RBAB_ADDBAND RI_KEY_TERMSRV_SHADOW
RBAB_AUTOSIZE RI_MOUSE_BUTTON_1_DOWN
RBBIM_BACKGROUND RI_MOUSE_BUTTON_1_UP
Constants

<table>
<thead>
<tr>
<th>Windows Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBBIM_CHILD</td>
<td>RI_MOUSE_BUTTON_2_DOWN</td>
</tr>
<tr>
<td>RBBIM_CHILDSIZE</td>
<td>RI_MOUSE_BUTTON_2_UP</td>
</tr>
<tr>
<td>RBBIM_COLORS</td>
<td>RI_MOUSE_BUTTON_3_DOWN</td>
</tr>
<tr>
<td>RBBIM_HEADERSIZE</td>
<td>RI_MOUSE_BUTTON_3_UP</td>
</tr>
<tr>
<td>RBBIM_ID</td>
<td>RI_MOUSE_BUTTON_4_DOWN</td>
</tr>
<tr>
<td>RBBIM_IDEALSIZ E</td>
<td>RI_MOUSE_BUTTON_4_UP</td>
</tr>
<tr>
<td>RBBIM_IMAGE</td>
<td>RI_MOUSE_BUTTON_5_DOWN</td>
</tr>
<tr>
<td>RBBIM_LPARAM</td>
<td>RI_MOUSE_BUTTON_5_UP</td>
</tr>
<tr>
<td>RBBIM_SIZE</td>
<td>RI_MOUSE_LEFT_BUTTON_DOWN</td>
</tr>
<tr>
<td>RBBIM_STYLE</td>
<td>RI_MOUSE_LEFT_BUTTON_UP</td>
</tr>
<tr>
<td>RBBIM_TEXT</td>
<td>RI_MOUSE_MIDDLE_BUTTON_DOWN</td>
</tr>
<tr>
<td>RBBIM_BREAK</td>
<td>RI_MOUSE_MIDDLE_BUTTON_UP</td>
</tr>
<tr>
<td>RBBIM_CHILDLEDGE</td>
<td>RI_MOUSE_RIGHT_BUTTON_DOWN</td>
</tr>
<tr>
<td>RBBS_FIXEDBMP</td>
<td>RI_MOUSE_RIGHT_BUTTON_UP</td>
</tr>
<tr>
<td>RBBS_FIXEDSIZE</td>
<td>RI_MOUSE_WHEEL</td>
</tr>
<tr>
<td>RBBS_GRIPPERALWAYS</td>
<td>RID_HEADER</td>
</tr>
<tr>
<td>RBBS_HIDDEN</td>
<td>RID_INPUT</td>
</tr>
<tr>
<td>RBBS_HIDETITLE</td>
<td>RIDEV_APPKEYS</td>
</tr>
<tr>
<td>RBBS_NOGRIPPER</td>
<td>RIDEV_CAPTUREMOUSE</td>
</tr>
<tr>
<td>RBBS_NOVERT</td>
<td>RIDEV_EXCLUDE</td>
</tr>
<tr>
<td>RBBS_TOPALIGN</td>
<td>RIDEV_EXMODEMASK</td>
</tr>
<tr>
<td>RBBS_USECHEVRON</td>
<td>RIDEV_INPUTSINK</td>
</tr>
<tr>
<td>RBBS_VARIABLEHEIGHT</td>
<td>RIDEV_NOHOTKEYS</td>
</tr>
<tr>
<td>RBHT_CAPTION</td>
<td>RIDEV_NOLEGACY</td>
</tr>
<tr>
<td>RBHT_CHEVRON</td>
<td>RIDEV_PAGEONLY</td>
</tr>
<tr>
<td>RBHT_CLIENT</td>
<td>RIDEV_REMOVE</td>
</tr>
<tr>
<td>RBHT_GRABBER</td>
<td>RIDI_DEVICEINFO</td>
</tr>
<tr>
<td>RBHT_NOWHERE</td>
<td>RIDI_DEVICENAME</td>
</tr>
<tr>
<td>RBIM_IMAGE</td>
<td>RIDI_PREPAREDDATA</td>
</tr>
<tr>
<td>RBN_AUTOBREAK</td>
<td>RIM_INPUT</td>
</tr>
<tr>
<td>RBN_AUTOSIZE</td>
<td>RIM_INPUTSINK</td>
</tr>
<tr>
<td>RBN_BEGINDRAG</td>
<td>RIM_TYPEHID</td>
</tr>
<tr>
<td>RBN_CHEVRONPUSHED</td>
<td>RIM_TYPEKEYBOARD</td>
</tr>
<tr>
<td>RBN_CHILDSIZE</td>
<td>RIM_TYPEMOUSE</td>
</tr>
<tr>
<td>RBN_DELETEDBAND</td>
<td>RUSSIAN_CHARSET</td>
</tr>
<tr>
<td>RBN_DELETINGBAND</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Windows Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVE_CTM</td>
<td>SPI_GETFILTERKEYS</td>
</tr>
<tr>
<td>SB_BOTH</td>
<td>SPI_GETFLATMENU</td>
</tr>
<tr>
<td>SB_BOTTOM</td>
<td>SPI_GETFOCUSBORDERHEIGHT</td>
</tr>
<tr>
<td>SB_CONST_ALPHA</td>
<td>SPI_GETFOCUSBORDERWIDTH</td>
</tr>
<tr>
<td>SB_CTL</td>
<td>SPI_GETFONTSMOOTHING</td>
</tr>
<tr>
<td>SB_ENDSCROLL</td>
<td>SPI_GETFONTSMOOTHINGCONTRAST</td>
</tr>
<tr>
<td>SB_GETBORDERS</td>
<td>SPI_GETFONTSMOOTHINGORIENTATION</td>
</tr>
<tr>
<td>SB_GetIcon</td>
<td>SPI_GetFontSmoothingType</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>SB_GetParts</td>
<td>SPI_GetForegroundFlashCount</td>
</tr>
<tr>
<td>SB_GetRect</td>
<td>SPI_GetForegroundLockTimeout</td>
</tr>
<tr>
<td>SB_GetText</td>
<td>SPI_GetGradientCaptions</td>
</tr>
<tr>
<td>SB_GetTextA</td>
<td>SPI_GetGridGranularity</td>
</tr>
<tr>
<td>SB_GetTextLength</td>
<td>SPI_GetHighContrast</td>
</tr>
<tr>
<td>SB_GetTextLengthA</td>
<td>SPI_GetHotTracking</td>
</tr>
<tr>
<td>SB_GetTextLengthW</td>
<td>SPI_GetIconMetrics</td>
</tr>
<tr>
<td>SB_GetTextW</td>
<td>SPI_GetIconTitleLogFont</td>
</tr>
<tr>
<td>SB_GetTipTextA</td>
<td>SPI_GetIconTitleWrap</td>
</tr>
<tr>
<td>SB_GetTipTextW</td>
<td>SPI_GetKeyboardCues</td>
</tr>
<tr>
<td>SB_GetUnicodeFormat</td>
<td>SPI_GetKeyboardDelay</td>
</tr>
<tr>
<td>SB_GetRect</td>
<td>SPI_GetKeyboardPREF</td>
</tr>
<tr>
<td>SB_Grad_Rect</td>
<td>SPI_GetKeyboardSpeed</td>
</tr>
<tr>
<td>SB_Grad_TRI</td>
<td>SPI_GetListBoxSmoothScrolling</td>
</tr>
<tr>
<td>SB_Horz</td>
<td>SPI_GetLowPowerActive</td>
</tr>
<tr>
<td>SB_ISSimple</td>
<td>SPI_GetLowPowerTimeout</td>
</tr>
<tr>
<td>SB_Left</td>
<td>SPI_GetMenuAnimation</td>
</tr>
<tr>
<td>SB_Linedown</td>
<td>SPI_GetMenuFade</td>
</tr>
<tr>
<td>SB_Lineleft</td>
<td>SPI_GetMenuShowDelay</td>
</tr>
<tr>
<td>SB_Lineright</td>
<td>SPI_GetMenuShowDelay</td>
</tr>
<tr>
<td>SB_Lineup</td>
<td>SPI_GetMenuShowDelay</td>
</tr>
<tr>
<td>SB_None</td>
<td>SPI_GetMenuUnderlines</td>
</tr>
<tr>
<td>SB_Pagedown</td>
<td>SPI_GetMinimizedMetrics</td>
</tr>
<tr>
<td>SB_Pageleft</td>
<td>SPI_GetMouse</td>
</tr>
<tr>
<td>SB_Pageright</td>
<td>SPI_GetMouseClickLock</td>
</tr>
<tr>
<td>SB_Pageup</td>
<td>SPI_GetMouseClickLockTime</td>
</tr>
<tr>
<td>SB_Pixel Alpha</td>
<td>SPI_GetMouseHoverHeight</td>
</tr>
<tr>
<td>SB_Premult Alpha</td>
<td>SPI_GetMouseHoverTime</td>
</tr>
<tr>
<td>SB_Right</td>
<td>SPI_GetMouseHoverWidth</td>
</tr>
<tr>
<td>SB_SetBkColor</td>
<td>SPI_GetMouseKeys</td>
</tr>
<tr>
<td>SB_SetIcon</td>
<td>SPI_GetMouseSonar</td>
</tr>
<tr>
<td>SB_SetMinHeight</td>
<td>SPI_GetMouseSpeed</td>
</tr>
<tr>
<td>SB_SetParts</td>
<td>SPI_GetMouse Trails</td>
</tr>
<tr>
<td>SB_SetText</td>
<td>SPI_GetMouseVanish</td>
</tr>
<tr>
<td>SB_SetTextA</td>
<td>SPI_GetNonClientMetrics</td>
</tr>
<tr>
<td>SB_SetTextW</td>
<td>SPI_GetPowerOffActive</td>
</tr>
<tr>
<td>SB_SetTipTextA</td>
<td>SPI_GetPowerOffTimeout</td>
</tr>
<tr>
<td>SB_SetTipTextW</td>
<td>SPI_GetScreenReader</td>
</tr>
<tr>
<td>SB_SetUnicodeFormat</td>
<td>SPI_GetScreenSaveActive</td>
</tr>
<tr>
<td>SB_Simple</td>
<td>SPI_GetScreenSaveRunning</td>
</tr>
<tr>
<td>SB_SimpleId</td>
<td>SPI_GetScreenSaveTimeout</td>
</tr>
<tr>
<td>SB_ThumbPosition</td>
<td>SPI_GetSelectionFade</td>
</tr>
<tr>
<td>SB_Thumbrack</td>
<td>SPI_GetSerialKeys</td>
</tr>
<tr>
<td>SB_Top</td>
<td>SPI_GetShowIMEUI</td>
</tr>
<tr>
<td>SB_Vert</td>
<td>SPI_GetShowSounds</td>
</tr>
<tr>
<td>SBars_SizeGrip</td>
<td>SPI_GetSnapToDefButton</td>
</tr>
</tbody>
</table>
SBARS_TOOLTIPS  SPI_GETSOUNDSENTRY
SBM_ENABLE_ARROWS  SPI_GETSTICKYKEYS
SBM_GETPOS  SPI_GETTOGGLEKEYS
SBM_GETRANGE  SPI_GETTOOLTIPANIMATION
SBM_GETSCROLLBARINFO  SPI_GETTOOLTIPFADE
SBM_GETSCROLLINFO  SPI_GETUIEFFECTS
SBM_SETPOS  SPI_GETWHEELSCROLLLINES
SBM_SETRANGE  SPI_GETWINDOWSEXTENSION
SBM_SETRANGEREDRAW  SPI_GETWORKAREA
SBM_SETSCROLLINFO  SPI_ICONHORIZONTALSPACING
SBN_SIMPLEMODECHANGE  SPI_ICONVERTICALSPACING
SBS_BOTTOMALIGN  SPI_LANGDRIVER
SBS_HORZ  SPI_SCREENSAVERRUNNING
SBS_LEFTALIGN  SPI_SETACCESTIMEOUT
SBS_RIGHTALIGN  SPI_SETACTIVEWINDOWTRACKING
SBS_SIZEBOX  SPI_SETACTIVEWNDTRKTIMEOUT
SBS_SIZEBOXBOTTOMRIGHTALIGN  SPI_SETACTIVEWNDTRKZORDER
SBS_SIZEBOXTOPLEFTALIGN  SPI_SETANIMATION
SBS_SIZEGRIP  SPI_SETBEEP
SBS_TOPALIGN  SPI_SETBLOCKSENDINPUTRESETS
SBS_VERT  SPI_SETBORDER
SBS_NOBORDERS  SPI_SETCARETWIDTH
SBT_NOTABPARSING  SPI_SETCOMBOBOXANIMATION
SBT_OWNERDRAW  SPI_SETCURSORS
SBT_POPOUT  SPI_SETCURSORSHADOW
SBT_RTLREADING  SPI_SETDEFAULTINPUTLANG
SBT_TOOLTIPS  SPI_SETDESKPATTERN
SC_ARRANGE  SPI_SETDESKWALLPAPER
SC_CLOSE  SPI_SETDOUBLECLICKTIME
SC_CONTEXTHELP  SPI_SETDOUBLECLKHEIGHT
SC_DEFAULT  SPI_SETDOUBLECLKWIDTH
SC_HOTKEY  SPI_SETDRAGFULLWINDOWS
SC_HSCROLL  SPI_SETDRAGHEIGHT
SC_ICON  SPI_SETDRAGWIDTH
SC_KEYMENU  SPI_SETDROPShadow
SC_MAXIMIZE  SPI_SETFASTTASKSWITCH
SC_MINIMIZE  SPI_SETFILTERKEYS
SC_MONITORPOWER  SPI_SETFLATMENU
SC_MOUSEMENU  SPI_SETFOCUSBORDERHEIGHT
SC_MOVE  SPI_SETFOCUSBORDERWIDTH
SC_NEXTWINDOW  SPI_SETFONTSMOOTHING
SC_PREVWINDOW  SPI_SETFONTSMOOTHINGCONTRAST
SC_RESTORE  SPI_SETFONTSMOOTHING ORIENTATION
SC_SCREENSAVE  SPI_SETFONTSMOOTHINGTYPE
SC_SEPARATOR  SPI_SETFOREGROUNDFLASHCOUNT
SC_SIZE  SPI_SETFOREGROUNDLOCKTIMEOUT
<table>
<thead>
<tr>
<th>Constants</th>
<th>SPI_SET...</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC_TASKLIST</td>
<td>SPI_SETGRADIENTCAPTIONS</td>
</tr>
<tr>
<td>SC_VSCROLL</td>
<td>SPI_SETGRIDGRANULARITY</td>
</tr>
<tr>
<td>SC_ZOOM</td>
<td>SPI_SETHANDHELD</td>
</tr>
<tr>
<td>SCALINGFACTORX</td>
<td>SPI_SETHIGHCONTRAST</td>
</tr>
<tr>
<td>SCALINGFACTORY</td>
<td>SPI_SETHOTTRACKING</td>
</tr>
<tr>
<td>SCREEN_FONTPROPERTIES</td>
<td>SPI_SETICONMETRICS</td>
</tr>
<tr>
<td>SELECTPAPERSOURCE</td>
<td>SPI_SETICONS</td>
</tr>
<tr>
<td>SERKF_AVAILABLE</td>
<td>SPI_SETICONTITLELOGFONT</td>
</tr>
<tr>
<td>SERKF_INDICATOR</td>
<td>SPI_SETICONTITLEWRAP</td>
</tr>
<tr>
<td>SERKF_SERIALKEYSON</td>
<td>SPI_SETKEYBOARDCUES</td>
</tr>
<tr>
<td>SET_ARC_DIRECTION</td>
<td>SPI_SETKEYBOARDDELAY</td>
</tr>
<tr>
<td>SET_BACKGROUND_COLOR</td>
<td>SPI_SETKEYBOARDPREF</td>
</tr>
<tr>
<td>SET_BOUNDS</td>
<td>SPI_SETKEYBOARDSPEED</td>
</tr>
<tr>
<td>SET_CLIP_BOX</td>
<td>SPI_SETLANGTOGGLE</td>
</tr>
<tr>
<td>SET_MIRROR_MODE</td>
<td>SPI_SETLISTBOXSMOOTHSCROLLING</td>
</tr>
<tr>
<td>SET_POLY_MODE</td>
<td>SPI_SETLOWPOWERACTIVE</td>
</tr>
<tr>
<td>SET_SCREEN_ANGLE</td>
<td>SPI_SETLOWPOWERTIMEOUT</td>
</tr>
<tr>
<td>SET_SPREAD</td>
<td>SPI_SETMENUANIMATION</td>
</tr>
<tr>
<td>SETABORTPROC</td>
<td>SPI_SETMENUDROPALIGNMENT</td>
</tr>
<tr>
<td>SETALLJUSTVALUES</td>
<td>SPI_SETMENUFADE</td>
</tr>
<tr>
<td>SETCHARSET</td>
<td>SPI_SETMENUHOWNUMERIC</td>
</tr>
<tr>
<td>SETCOLORTABLE</td>
<td>SPI_SETMENUUNDERLINES</td>
</tr>
<tr>
<td>SETCOPYCOUNT</td>
<td>SPI_SETMINIMIZEDMETRICS</td>
</tr>
<tr>
<td>SETDIBSCALING</td>
<td>SPI_SETMOUSE</td>
</tr>
<tr>
<td>SETICMPROFILE_EMBEDDED</td>
<td>SPI_SETMOUSEBUTTONSWAP</td>
</tr>
<tr>
<td>SETKERNTRACK</td>
<td>SPI_SETMOUSECLICKLOCK</td>
</tr>
<tr>
<td>SETLINECAP</td>
<td>SPI_SETMOUSECLICKLOCKTIME</td>
</tr>
<tr>
<td>SETLINEJOIN</td>
<td>SPI_SETMOUSEGOVERHEIGHT</td>
</tr>
<tr>
<td>SETMETERLIMIT</td>
<td>SPI_SETMOUSEGEOVERWIDTH</td>
</tr>
<tr>
<td>SHADEBLENDCAPS</td>
<td>SPI_SETMOUSEGOVERTIME</td>
</tr>
<tr>
<td>SHIFTJS_CHARSET</td>
<td>SPI_SETMOUSEKEYS</td>
</tr>
<tr>
<td>SHOW_FULLSCREEN</td>
<td>SPI_SETMOUSESONAR</td>
</tr>
<tr>
<td>SHOW_ICONWINDOW</td>
<td>SPI_SETMOUSESPEED</td>
</tr>
<tr>
<td>SHOW_OPENNOACTIVATE</td>
<td>SPI_SETMOUSESTRAILS</td>
</tr>
<tr>
<td>SHOW_OPENWINDOW</td>
<td>SPI_SETMOUSEEVANISH</td>
</tr>
<tr>
<td>SIF_ALL</td>
<td>SPI_SETNONCLIENTMETRICS</td>
</tr>
<tr>
<td>SIF_DISABLENOSCROLL</td>
<td>SPI_SETPENWINDOWS</td>
</tr>
<tr>
<td>SIF_PAGE</td>
<td>SPI_SETPOWEROFFACTIVE</td>
</tr>
<tr>
<td>SIF_POS</td>
<td>SPI_SETPOWEROFFTIMEOUT</td>
</tr>
<tr>
<td>SIF_RANGE</td>
<td>SPI_SETSCREENREADER</td>
</tr>
<tr>
<td>SIF_TRACKPOS</td>
<td>SPI_SETSCREENSAVEACTIVE</td>
</tr>
<tr>
<td>SIMPLEREGION</td>
<td>SPI_SETSCREENSAVERUNNING</td>
</tr>
<tr>
<td>SIMULATED_FONTPROPERTIES</td>
<td>SPI_SETSCREENSAVETIMEOUT</td>
</tr>
<tr>
<td>SIZE_MAXHIDE</td>
<td>SPI_SETSELECTIONFADE</td>
</tr>
<tr>
<td>SIZE_MAXIMIZED</td>
<td>SPI_SETSEIZEDEALKEYS</td>
</tr>
<tr>
<td>SIZE_MAXSHOW</td>
<td>SPI_SETSHOWIMEUI</td>
</tr>
<tr>
<td>Constants</td>
<td>Values</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>SIZE_MINIMIZED</td>
<td>SPI_SETSHOWSOUNDS</td>
</tr>
<tr>
<td>SIZE_RESTORED</td>
<td>SPI_SETSNAPTOODEFBUTTON</td>
</tr>
<tr>
<td>SIZEFULLSCREEN</td>
<td>SPI_SETSOUNDSENTRY</td>
</tr>
<tr>
<td>SIZEICONIC</td>
<td>SPI_SETSTICKYKEYS</td>
</tr>
<tr>
<td>SIZENORMAL</td>
<td>SPI_SETTOGGLEKEYS</td>
</tr>
<tr>
<td>SIZEPALETTE</td>
<td>SPI_SETTOOLTIPANIMATION</td>
</tr>
<tr>
<td>SIZEZOOMHIDE</td>
<td>SPI_SETTOOLTIPFADE</td>
</tr>
<tr>
<td>SIZEZOOMSHOW</td>
<td>SPI_SETUIEFFECTS</td>
</tr>
<tr>
<td>SKF_AUDIBLEFEEDBACK</td>
<td>SPI_SETWHEELSCROLLLINES</td>
</tr>
<tr>
<td>SKF_AVAILABLE</td>
<td>SPI_SETWORKAREA</td>
</tr>
<tr>
<td>SKF_CONFIRMHOTKEY</td>
<td>SPIF_SENDCHANGE</td>
</tr>
<tr>
<td>SKF_HOTKEYACTIVE</td>
<td>SPIF_SENDWININICHAANGE</td>
</tr>
<tr>
<td>SKF_HOTKEYSOUND</td>
<td>SPIF_UPDATEINIFILE</td>
</tr>
<tr>
<td>SKF_INDICATOR</td>
<td>SS_BITMAP</td>
</tr>
<tr>
<td>SKF_LALTTLATCHED</td>
<td>SS_BLACKFRAME</td>
</tr>
<tr>
<td>SKF_LALTLOCKED</td>
<td>SS_BLACKRECT</td>
</tr>
<tr>
<td>SKF_LCTLTLATCHED</td>
<td>SS_CENTER</td>
</tr>
<tr>
<td>SKF_LCTLLATCHED</td>
<td>SS_CENTERIMAGE</td>
</tr>
<tr>
<td>SKF_LSHIFTLATCHED</td>
<td>SS_EDITCONTROL</td>
</tr>
<tr>
<td>SKF_LSHIFTLOCKED</td>
<td>SS_ELLIPSIMASK</td>
</tr>
<tr>
<td>SKF_LWINLATCHED</td>
<td>SS_ENDELLIPSISS</td>
</tr>
<tr>
<td>SKF_LWINLOCKED</td>
<td>SS_ENHMETAFILE</td>
</tr>
<tr>
<td>SKF_RALTTLATCHED</td>
<td>SS_ETCHEDFRAME</td>
</tr>
<tr>
<td>SKF_RALTLOCKED</td>
<td>SS_ETCHEDHORIZ</td>
</tr>
<tr>
<td>SKF_RCTLLATCHED</td>
<td>SS_ETCHEDVERT</td>
</tr>
<tr>
<td>SKF_RCTLLocked</td>
<td>SS_GRAYFRAME</td>
</tr>
<tr>
<td>SKF_RSHIFTLATCHED</td>
<td>SS_GRAYRECT</td>
</tr>
<tr>
<td>SKF_RSHIFTLOCKED</td>
<td>SS_ICON</td>
</tr>
<tr>
<td>SKF_RWINLATCHED</td>
<td>SS_LEFT</td>
</tr>
<tr>
<td>SKF_RWINLOCKED</td>
<td>SS_LEFTNOWORDWRAP</td>
</tr>
<tr>
<td>SKF_STICKYKEYSON</td>
<td>SS_NOPREFIX</td>
</tr>
<tr>
<td>SKF_TRISTATE</td>
<td>SS_NOTIFY</td>
</tr>
<tr>
<td>SKF_TWOKEYSOFF</td>
<td>SS_OWNERDRAW</td>
</tr>
<tr>
<td>SLE_ERROR</td>
<td>SS_PATHELLIPSIS</td>
</tr>
<tr>
<td>SLE_MINORERROR</td>
<td>SS_REALSIZECONTROL</td>
</tr>
<tr>
<td>SLE_WARNING</td>
<td>SS_REALSIZEIMAGE</td>
</tr>
<tr>
<td>SM_ARRANGE</td>
<td>SS_RIGHT</td>
</tr>
<tr>
<td>SM_CLEANBOOT</td>
<td>SS_RIGHTJUST</td>
</tr>
<tr>
<td>SM_CMETRICS</td>
<td>SS_SIMPLE</td>
</tr>
<tr>
<td>SM_CMONITORS</td>
<td>SS_SUNKEN</td>
</tr>
<tr>
<td>SM_CMOMOUSEBUTTONS</td>
<td>SS_TYPENAME</td>
</tr>
<tr>
<td>SM_CXBORDER</td>
<td>SS_USERITEM</td>
</tr>
<tr>
<td>SM_CXCURSOR</td>
<td>SS_WHITEFRAME</td>
</tr>
<tr>
<td>SM_CXDLGFRAME</td>
<td>SS_WHITERECT</td>
</tr>
<tr>
<td>SM_CXDUBLECLK</td>
<td>SS_WORDELLIPSISS</td>
</tr>
<tr>
<td>SM_CXDRAG</td>
<td>SSSF_AVAILABLE</td>
</tr>
</tbody>
</table>
SM_CXEDGE  SSF_INDICATOR
SM_CXFIXEDFRAME  SSF_SOUNDSENTRYON
SM_CXFOCUSBORDER  SSGF_DISPLAY
SM_CXFRAME  SSGF_NONE
SM_CXFULLSCREEN  SSTF_BORDER
SM_CXHSCROLL  SSTF_CHARS
SM_CXHTHUMB  SSTF_DISPLAY
SM_CXICON  SSTF_NONE
SM_CXICONSPACING  SSWF_CUSTOM
SM_CXMENUSIZE  SSWF_DISPLAY
SM_CXMIN  START_PAGE_GENERAL
SM_CXMINIMIZED  STARTDOC
SM_CXMINSIZING  STATE_SYSTEM_ALERT_HIGH
SM_CXMNTRACK  STATE_SYSTEM_ALERT_LOW
SM_CXSCREEN  STATE_SYSTEM_ALERT_MEDIUM
SM_CXSIZE  STATE_SYSTEM_ANIMATED
SM_CXSIZEFRAME  STATE_SYSTEM_BUSY
SM_CXSMICON  STATE_SYSTEM_CHECKED
SM_CXSMNSIZE  STATE_SYSTEM_COLLAPSED
SM_CXSVIRTUALSCREEN  STATE_SYSTEM_DEFAULT
SM_CXVSCROLL  STATE_SYSTEM_EXPANDED
SM_CYBORDER  STATE_SYSTEM_EXTSELECTABLE
SM_CYCAPTION  STATE_SYSTEM_FLOATING
SM_CYCURSOR  STATE_SYSTEM_FOCUSABLE
SM_CYDLGFRAME  STATE_SYSTEM_FOCUSED
SM_CYDOUBLECLK  STATE_SYSTEM_HOTTRACKED
SM_CYDRAG  STATE_SYSTEM_INDETERMINATE
SM_CYEDGE  STATE_SYSTEM_INVISIBLE
SM_CFIXEDFRAME  STATE_SYSTEM_LINKED
SM_CYFOCUSBORDER  STATE_SYSTEM_MAQUUED
SM_CYFRAME  STATE_SYSTEM_MIXED
SM_CYFULLSCREEN  STATE_SYSTEM_MOVEABLE
SM_CYHSCROLL  STATE_SYSTEM_MULTISELECTABLE
SM_CYICON  STATE_SYSTEM_OFFSCREEN
SM_CYICONSPACING  STATE_SYSTEM_PRESSED
SM_CYKANJIWINDOW  STATE_SYSTEM_PROTECTED
SM_CYMAXIMIZED  STATE_SYSTEM_READONLY
SM_CYMAXTRACK  STATE_SYSTEM_SELECTABLE
SM_CYMENU  STATE_SYSTEM_SELECTION
SM_CYMENUCHECK  STATE_SYSTEM_SELFVOICING
SM_CYMENUSIZE  STATE_SYSTEM_SIZEABLE
SM_CYMIN  STATE_SYSTEM_TRAVERSED
SM_CYMINIMIZED  STATE_SYSTEM_UNAVAILABLE
SM_CYMINSPIXING  STATE_SYSTEM_VALID
SM_CYMINTRACK  STD_COPY
SM_CYSCREEN  STD_CUT
SM_CYSIZE  STD_DELETE
SM_CYSIZEFRAME  STD_FILENEW
SM_CYSMCAPTION  STD_FILEOPEN
SM_CYSMICON  STD_FILESAVE
SM_CYSMSIZE  STD_FIND
SM_CYVIRTUALSCREEN  STD_HELP
SM_CYVSCROLL  STD_PASTE
SM_CYVTHUMB  STD_PRINT
SM_DBCSENABLED  STD_PRINTPRE
SM_DEBUG  STD_PROPERTIES
SM_IMMENABLED  STD_REDO
SM_MEDIACENTER  STD_REPLACE
SM_MENUDROPAIGNMENT  STD_UNDO
SM_MIDEASTENABLED  STM_GETICON
SM_MOUSEPRESEN  STM_GETIMAGE
SM_MOUSEWHEELPRESEN  STM_MSMAX
SM_NETWORK  STM_SETICON
SM_PENWINDOWS  STM_SETIMAGE
SM_REMOTECONTROL  STN_CLICKED
SM_REMOTESESSION  STN_DOUBLECLK
SM_RESERVED1  STN_DISABLE
SM_RESERVED2  STN_ENABLE
SM_RESERVED3  STK_LAST
SM_RESERVED4  STRETCH_ANDSCANS
SM_SAMEDISPLAYFORMAT  STRETCH_DELETESCANS
SM_SECURE  STRETCH_HALFTONE
SM_SHOWSOUNDS  STRETCH_ORSCANS
SM_SHUTTINDOWN  STRETCHBLT
SM_SLOWMACHINE  STRICT
SM_SWAPBUTTON  SW_ERASE
SM_TABLETPC  SW_FORCEMINIMIZE
SM_XVIRTUALSCREEN  SW_HIDE
SM_YVIRTUALSCREEN  SW_INVALIDATE
SMTO_ABORTIFHUNG  SW_MAX
SMTO_BLOCK  SW_MAXIMIZE
SMTO_NORMAL  SW_MINIMIZE
SMTO_NOTIMEOUTIFNOTHING  SW_NORMAL
SOUND_SYSTEM_APPEND  SW_OHERUNZOOM
SOUND_SYSTEM_APPSTART  SW_OHERZOOM
SOUND_SYSTEM_BEEP  SW_PARENTCLOSING
SOUND_SYSTEM_ERROR  SW_PARENTOPENING
SOUND_SYSTEM_FAULT  SW_RESTORE
SOUND_SYSTEM_INFORMATION  SW_SCROLLCHILDREN
<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOUND_SYSTEM_MAXIMIZE</td>
<td>SW_SHOW</td>
</tr>
<tr>
<td>SOUND_SYSTEM_MENUCOMMAND</td>
<td>SW_SHOWDEFAULT</td>
</tr>
<tr>
<td>SOUND_SYSTEM_MENUPopup</td>
<td>SW_SHOWMAXIMIZED</td>
</tr>
<tr>
<td>SOUND_SYSTEM_MINIMIZE</td>
<td>SW_SHOWMINIMIZED</td>
</tr>
<tr>
<td>SOUND_SYSTEM_QUESTION</td>
<td>SW_SHOWNA</td>
</tr>
<tr>
<td>SOUND_SYSTEM_RESTOREDOWN</td>
<td>SW_SHOWNOACTIVATE</td>
</tr>
<tr>
<td>SOUND_SYSTEM_RESTOREUP</td>
<td>SW_SHOWNOACTIVATE</td>
</tr>
<tr>
<td>SOUND_SYSTEM_SHUTDOWN</td>
<td>SW_SHOWNORMAL</td>
</tr>
<tr>
<td>SOUND_SYSTEM_STARTUP</td>
<td>SW_SMOOTHSCROLL</td>
</tr>
<tr>
<td>SOUND_SYSTEM_WARNING</td>
<td>SWP_ASYNCWINDOWPOS</td>
</tr>
<tr>
<td>SP_APPABORT</td>
<td>SWP_DEFERERASE</td>
</tr>
<tr>
<td>SP_ERROR</td>
<td>SWP_DRAWFRAME</td>
</tr>
<tr>
<td>SP_NOTREPORTED</td>
<td>SWP_FRAMECHANGED</td>
</tr>
<tr>
<td>SP_OUTOFDISK</td>
<td>SPW_HIDEWINDOW</td>
</tr>
<tr>
<td>SP_OUTOFMEMORY</td>
<td>SPW_NOACTIVATE</td>
</tr>
<tr>
<td>SP_USERABORT</td>
<td>SPW_NOCOPYBITS</td>
</tr>
<tr>
<td>SPCLPASSTHROUGH2</td>
<td>SWP_NOMOVE</td>
</tr>
<tr>
<td>SPI_GETACCESTIMEOUT</td>
<td>SPW_NOOWNERZORDER</td>
</tr>
<tr>
<td>SPI_GETACTIVEWINDOWTRACKING</td>
<td>SWP_NOREDRAW</td>
</tr>
<tr>
<td>SPI_GETACTIVEWNDTRKTIMEOUT</td>
<td>SWP_NOREPOSITION</td>
</tr>
<tr>
<td>SPI_GETACTIVEWNDTRKZORDER</td>
<td>SWP_NOSENDCHANGING</td>
</tr>
<tr>
<td>SPI_GETANIMATION</td>
<td>SWP_NOSIZE</td>
</tr>
<tr>
<td>SPI_GETBEEP</td>
<td>SWP_NOZORDER</td>
</tr>
<tr>
<td>SPI_GETBLOCKSENDINPUTRESETS</td>
<td>SWP_SHOWWINDOW</td>
</tr>
<tr>
<td>SPI_GETBORDER</td>
<td>SYMBOL_CHARSET</td>
</tr>
<tr>
<td>SPI_GETCARETWIDTH</td>
<td>SYSPAL_ERROR</td>
</tr>
<tr>
<td>SPI_GETCOMBOBOXANIMATION</td>
<td>SYSPAL_NOSTATIC</td>
</tr>
<tr>
<td>SPI_GETCURSORSHADOW</td>
<td>SYSPAL_NOSTATIC256</td>
</tr>
<tr>
<td>SPI_GETDEFAULTINPUTLANG</td>
<td>SYSPAL_STATIC</td>
</tr>
<tr>
<td>SPI_GETDESKWALLPAPER</td>
<td>SYSRGN</td>
</tr>
<tr>
<td>SPI_GETDRAGFULLWINDOWS</td>
<td>SYSTEM_FIXED_FONT</td>
</tr>
<tr>
<td>SPI_GETDROPSHADOW</td>
<td>SYSTEM_FONT</td>
</tr>
<tr>
<td>SPI_GETFASTTASKSWITCH</td>
<td></td>
</tr>
<tr>
<td>TA_BASELINE</td>
<td>TCN_KEYDOWN</td>
</tr>
<tr>
<td>TA_BOTTOM</td>
<td>TCN_SELCHANGE</td>
</tr>
<tr>
<td>TA_CENTER</td>
<td>TCN_SELCHANGING</td>
</tr>
<tr>
<td>TA_LEFT</td>
<td>TCS_BOTTOM</td>
</tr>
<tr>
<td>TA_MASK</td>
<td>TCS_BUTTONS</td>
</tr>
<tr>
<td>TA_NOUPDATECP</td>
<td>TCS_EX_FLATSEPARATORS</td>
</tr>
<tr>
<td>TA_RIGHT</td>
<td>TCS_EX_REGISTERSDROP</td>
</tr>
<tr>
<td>TA_RTLREADING</td>
<td>TCS_FIXEDWIDTH</td>
</tr>
<tr>
<td>TA_TOP</td>
<td>TCS_FLATBUTTONS</td>
</tr>
<tr>
<td>TA_UPDATECP</td>
<td>TCS_FOCUSNEVER</td>
</tr>
</tbody>
</table>
TB_ADDBITMAP TCS_FOCUSONBUTTONDOWN
TB_ADDBUTTONS TCS_FORCEICONLEFT
TB_ADDBUTTONSA TCS_FORCELABELLEFT
TB_ADDBUTTONSW TCS_HOTTRACK
TB_ADDSTRING TCS_MULTILINE
TB_ADDSTRINGA TCS_MULTISELECT
TB_ADDSTRINGW TCS_OWNERDRAWFIXED
TB_AUTOSIZE TCS_RAGGEDRIGHT
TB_BOTTOM TCS_RIGHT
TB_BUTTONCOUNT TCS_RIGHTJUSTIFY
TB_BUTTONSTRUCTSIZE TCS_SCROLLOPPOSITE
TB_CHANGEBITMAP TCS_SINGLELINE
TB_CHECKBUTTON TCS_TABS
TB_COMMANDTOINDEX TCS_TOOLTIP
TB_CUSTOMIZE TECHNOLOGY
TB_DELETEBUTTON TEXTCAPS
TB_ENABLEBUTTON THAI_CHARSET
TB_ENDTRACK TKF_AVAILABLE
TB_GETANCHORHIGHLIGHT TKF_CONFIRMHOTKEY
TB_GETBITMAP TKF_HOTKEYACTIVE
TB_GETBUTTON TKF_HOTKEYSOUND
TB_GETBUTTONINFO TKF_INDICATOR
TB_GETBUTTONINFOA TKF_TOGGLEKEYSON
TB_GETBUTTONINFOW TM_MULTICODEPAGE
TB_GETBUTTONSIZE TM_MULTILEVELUNDO
TB_GETBUTTONTEXT TM_PLAINTEXT
TB_GETBUTTONTEXTA TM_RICHTEXT
TB_GETBUTTONTEXTW TM_SINGLECODEPAGE
TB_GETCOLORSCHEME TM_SINGLELEVELUNDO
TB_GETDISABLEDIMAGELIST TME_CANCEL
TB_GETEXTENDEDSTYLE TME_HOVER
TB_GETHOTIMAGELIST TME_LEAVE
TB_GETHOTITEM TME_NONCLIENT
TB_GETIMAGELIST TME_QUERY
TB_GETINSERTMARK TMPF_DEVICE
TB_GETINSERTMARKCOLOR TMPF_FIXED_PITCH
TB_GETITEMRECT TMPF_TRUETYPE
TB_GETMAXSIZE TMPF_VECTOR
TB_GETMETRICS TPM_BOTTOMALIGN
TB_GETOBJECT TPM_CENTERALIGN
TB_GETPADDING TPM_HORNEGANIMATION
TB_GETRECT TPM_HORIZONAL
TB_GETRECT TMP_HORNEGOSANIMATION
TB_GETROWS TMP_HORPOSANIMATION
TB_GETSTATE TMP_LAYOUTRTL
TB_GETSTRING TMP_LEFTALIGN
TB_GETSTRINGA  TPM_LEFTBUTTON
TB_GETSTRINGW  TPM_NOANIMATION
TB_GETSTYLE  TPM_NONOTIFY
TB_GETTEXTROWS  TPM_RECURSE
TB_GETTOOLTIP  TPM_RETURNCMD
TB_GETUNICODEFORMAT  TPM_RIGHTALIGN
TB_HIDEBUTTON  TPM_RIGHTBUTTON
TB_HITTEST  TPM_TOPALIGN
TB_INDETERMINATE  TPM_VCENTERALIGN
TB_INSERTBUTTON  TPM_VERNEGANIMATION
TB_INSERTBUTTONA  TPM_VERTANIMATION
TB_INSERTBUTTONW  TPM_VERTICAL
TB_INSERTMARKHITTEST  TRANSFORM_CTM
TB_ISBUTTONCHECKED  TRANSFER
TB_ISBUTTONENABLED  TRUE
TB_ISBUTTONHIDDEN  TRUETYPE_BOLD
TB_ISBUTTONHIGHLIGHTED  TT_AVAILABLE
TB_ISBUTTONINDETERMINATE  TT_ENABLED
TB_ISBUTTONPRESSED  TT_POLYGON_TYPE
TB_LINEDOWN  TT_PRIM_CSPLINE
TB_LINEUP  TT_PRIM_LINE
TB_LOADIMAGES  TT_PRIM_OSPLINE
TB_MAPACCELERATOR  TTDT_AUTOMATIC
TB_MAPACCELERATORA  TTDT_AUTOPOP
TB_MAPACCELERATORW  TTDT_INITIAL
TB_MARKBUTTON  TTDT_RESHOW
TB_MOVEBUTTON  TTF_ABSOLUTE
TB_PAGEDOWN  TTF_CENTER_TIP
TB_PAGEUP  TTF_DI_SETITEM
TB_PRESSBUTTON  TTF_IDISHWND
TB_REPLACEBITMAP  TTF_PARSELINKS
TB_SAVERESTORE  TTF_RTLREADING
TB_SAVERESTOREA  TTF_SUBCLASS
TB_SAVERESTOREW  TTF_TRACK
TB_SETANCHORHIGHLIGHT  TTF_TRANSPARENT
TB_SETBITMAPSIZE  TTI_ERROR
TB_SETBUTTONINFO  TTI_INFO
TB_SETBUTTONINFOA  TTI_NONE
TB_SETBUTTONINFOW  TTI_WARNING
TB_SETBUTTONSIZE  TTM_ACTIVATE
TB_SETBUTTONWIDTH  TTM_ADDTOOL
TB_SETCMDID  TTM_ADDTOOLA
TB_SETCOLORSCHEME  TTM_ADDTOOLW
TB_SETDISABLEDIMAGELIST  TTM_ADJUSTRECT
TB_SETDRAWTEXTFLAGS  TTM_DELTOOL
TB_SETEXTENDEDSTYLE  TTM_DELTOOL
TB_SETHOTIMAGELIST  TTM_DELTOOLW
TB_SETHOTITEM     TTM_ENUMTOOLS
TB_SETIMAGELIST   TTM_ENUMTOOLSA
TB_SETINDENT      TTM_GETBUBBLESIZE
TB_SETINSERTMARK  TTM_GETCURRENTTOOL
TB_SETINSERTMARKCOLOR TTM_GETCURRENTTOOLA
TB_SETMAXTEXTROWS TTM_GETCURRENTTOOLW
TB_SETMETRICS     TTM_GETDELAYTIME
TB_SETPADDING     TTM_GETMARGIN
TB_SETPARENT      TTM_GETMAXTIPWIDTH
TB_SETROWS        TTM_GETTEXT
TB_SETSTATE       TTM_GETTEXTA
TB_SETSTYLE       TTM_GETTEXTW
TB_SETTOOLTIPS    TTM_HITTEST
TB_SETUNICODEFORMAT TTM_HITTESTA
TB_SETWINDOWTHEME TTM_HITTESTW
TB_THUMBPOSITION  TTM_HITTESTA
TB_THUMBTRACK     TTM_HITTESTW
TB_TOP            TTM_HITTEST
TBBF_LARGE        TTM_HITTEST
TBCD_CHANNEL      TTM_HITTEST
TBCD_THUMB        TTM_HITTEST
TBCD_TICS         TTM_HITTESTA
TBCDRF_BLENDICON  TTM_HITTESTW
TBCDRF_HILITEHOTTRACK TTM_HITTEST
TBCDRF_NOBACKGROUND TTM_HITTEST
TBCDRF_NOEDGES   TTM_HITTEST
TBCDRF_NOETCHEDEFFECT TTM_HITTESTA
TBCDRF_NOMARK    TTM_HITTESTW
TBCDRF_NOOFFSET  TTM_HITTEST
TBDDRET_DEFAULT  TTM_HITTEST
TBDDRET_NODEFAULT TTM_HITTESTA
TBDDRET_TREATPRESSED TTM_HITTESTW
TBIF_BYINDEX     TTM_HITTEST
TBIF_COMMAND     TTM_HITTESTA
TBIF_IMAGE       TTM_HITTESTW
TBIF_LPARAM      TTM_HITTEST
TBIF_SIZE        TTM_HITTEST
TBIF_STATE       TTM_HITTESTA
TBIF_STYLE       TTM_HITTESTW
TBIF_TEXT        TTM_HITTEST
TBIMHT_AFTER     TTM_HITTEST
TBIMHT_BACKGROUND TTM_HITTEST
TBM_CLEARSEL     TTM_HITTESTA
TBM_CLEARTICS    TTM_HITTESTW
TBM_GETBUDDY     TTM_HITTEST

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBN_GETDISPINFO</td>
<td>TVHT_ONITEMLABEL</td>
</tr>
<tr>
<td>TBN_GETDISPINFOA</td>
<td>TVHT_ONITEMRIGHT</td>
</tr>
<tr>
<td>TBN_GETDISPINFOW</td>
<td>TVHT_ONITEMSTATEICON</td>
</tr>
<tr>
<td>TBN_GETINFOTIP</td>
<td>TVHT_TOOLEFT</td>
</tr>
<tr>
<td>TBN_GETINFOTIPA</td>
<td>TVHT_TORIGHT</td>
</tr>
<tr>
<td>TBN_GETINFOTIPW</td>
<td>TVIF_CHILDREN</td>
</tr>
<tr>
<td>TBN_GETOBJECT</td>
<td>TVIF_DI_SETITEM</td>
</tr>
<tr>
<td>TBN_HOTITEMCHANGE</td>
<td>TVIFHANDLE</td>
</tr>
<tr>
<td>TBN_INITCUSTOMIZE</td>
<td>TVIF_IMAGE</td>
</tr>
<tr>
<td>TBN_QUERYDELETE</td>
<td>TVIF_INTEGRAL</td>
</tr>
<tr>
<td>TBN_QUERYINSERT</td>
<td>TVIF_PARAM</td>
</tr>
<tr>
<td>TBN_RESET</td>
<td>TVIF_SELECTEDIMAGE</td>
</tr>
<tr>
<td>TBN_RESTORE</td>
<td>TVIF_STATE</td>
</tr>
<tr>
<td>TBN_SAVE</td>
<td>TVIF_TEXT</td>
</tr>
<tr>
<td>TBN_TOOLBARCHANGE</td>
<td>TVIS_BOLD</td>
</tr>
<tr>
<td>TBNF_DI_SETITEM</td>
<td>TVIS_CUT</td>
</tr>
<tr>
<td>TBNF_IMAGE</td>
<td>TVIS_DROPHILITED</td>
</tr>
<tr>
<td>TBNF_TEXT</td>
<td>TVIS_EXPANDED</td>
</tr>
<tr>
<td>TBNRF_ENDCUSTOMIZE</td>
<td>TVIS_EXPANDEDONCE</td>
</tr>
<tr>
<td>TBNRF_HIDEHELP</td>
<td>TVIS_EXPANDPARTIAL</td>
</tr>
<tr>
<td>TBS_AUTOTICKS</td>
<td>TVIS_OVERLAYMASK</td>
</tr>
<tr>
<td>TBS_BOTH</td>
<td>TVIS_SELECTED</td>
</tr>
<tr>
<td>TBS_BOTTOM</td>
<td>TVIS_STATEIMAGEMASK</td>
</tr>
<tr>
<td>TBS_DOWNISLEFT</td>
<td>TVIS_USERMASK</td>
</tr>
<tr>
<td>TBS_ENABLESELRANGE</td>
<td>TVM_CREATEDRAGIMAGE</td>
</tr>
<tr>
<td>TBS_FIXEDLENGTH</td>
<td>TVM_DELETEITEM</td>
</tr>
<tr>
<td>TBS_HORZ</td>
<td>TVM_EDITLABEL</td>
</tr>
<tr>
<td>TBS_LEFT</td>
<td>TVM_EDITLABELA</td>
</tr>
<tr>
<td>TBS_NOTHUMB</td>
<td>TVM_EDITLABELW</td>
</tr>
<tr>
<td>TBS_NOTICKS</td>
<td>TVM_ENDEDITLABELNOW</td>
</tr>
<tr>
<td>TBS_REVERSED</td>
<td>TVM_ENSUREVISIBLE</td>
</tr>
<tr>
<td>TBS_RIGHT</td>
<td>TVM_EXPAND</td>
</tr>
<tr>
<td>TBS_TOOLTIPS</td>
<td>TVM_GETBKCOLOR</td>
</tr>
<tr>
<td>TBS_TOP</td>
<td>TVM_GETCOUNT</td>
</tr>
<tr>
<td>TBS_VERT</td>
<td>TVM_GETEDITCONTROL</td>
</tr>
<tr>
<td>TBSTATE_CHECKED</td>
<td>TVM_GETIMAGELIST</td>
</tr>
<tr>
<td>TBSTATE_ELLIPSES</td>
<td>TVM_GETINDENT</td>
</tr>
<tr>
<td>TBSTATE_ENABLED</td>
<td>TVM_GETINSERTMARKCOLOR</td>
</tr>
<tr>
<td>TBSTATE_HIDDEN</td>
<td>TVM_GETSEARCHSTRING</td>
</tr>
<tr>
<td>TBSTATE_INDETERMINATE</td>
<td>TVM_GETSEARCHSTRINGA</td>
</tr>
<tr>
<td>TBSTATE_MARKED</td>
<td>TVM_GETSEARCHSTRINGW</td>
</tr>
<tr>
<td>TBSTATE_PRESSED</td>
<td>TVM_GETITEM</td>
</tr>
<tr>
<td>TBSTATE_WRAP</td>
<td>TVM_GETITEMA</td>
</tr>
<tr>
<td>TBSTYLE_ALTDRAG</td>
<td>TVM_GETITEMHEIGHT</td>
</tr>
<tr>
<td>TBSTYLE_AUTOSIZE</td>
<td>TVM_GETITEMRECT</td>
</tr>
<tr>
<td>TBSTYLE_BUTTON</td>
<td>TVM_GETITEMSTATE</td>
</tr>
<tr>
<td>Constants</td>
<td>Function</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>TBSTYLE_CHECK</td>
<td>TVM_GETITEMW</td>
</tr>
<tr>
<td>TBSTYLE_CHECKGROUP</td>
<td>TVM_GETLINECOLOR</td>
</tr>
<tr>
<td>TBSTYLE_CUSTOMERASE</td>
<td>TVM_GETNEXTITEM</td>
</tr>
<tr>
<td>TBSTYLE_DROPDOWN</td>
<td>TVM_GETSCROLLTIME</td>
</tr>
<tr>
<td>TBSTYLE_EX_DOUBLEBUFFER</td>
<td>TVM_GETTEXTCOLOR</td>
</tr>
<tr>
<td>TBSTYLE_EX_DRAWDDARROWS</td>
<td>TVM_GETTOOLTIPS</td>
</tr>
<tr>
<td>TBSTYLE_EX_HIDECLIPPEDBUTTONS</td>
<td>TVM_GETUNICODEDEFORMATTED</td>
</tr>
<tr>
<td>TBSTYLE_EX_MIXEDBUTTONS</td>
<td>TVM_GETVISIBLECOUNT</td>
</tr>
<tr>
<td>TBSTYLE_FLAT</td>
<td>TVM_HITTEST</td>
</tr>
<tr>
<td>TBSTYLE_GROUP</td>
<td>TVM_INSERTITEM</td>
</tr>
<tr>
<td>TBSTYLE_LIST</td>
<td>TVM_INSERTITEMA</td>
</tr>
<tr>
<td>TBSTYLE_NOPREFIX</td>
<td>TVM_INSERTITEMW</td>
</tr>
<tr>
<td>TBSTYLE_REGISTERDROP</td>
<td>TVM_MAPACCIDTOTHREEITEM</td>
</tr>
<tr>
<td>TBSTYLE_SEP</td>
<td>TVM_MAPHTREEITEMTOACCID</td>
</tr>
<tr>
<td>TBSTYLE_TOOLTIPS</td>
<td>TVM_SELECTITEM</td>
</tr>
<tr>
<td>TBSTYLE_TRANSPARENT</td>
<td>TVM_SETBKCOLOR</td>
</tr>
<tr>
<td>TBSTYLE_WRAPABLE</td>
<td>TVM_SETIMAGELIST</td>
</tr>
<tr>
<td>TBTS_BOTTOM</td>
<td>TVM_SETINDENT</td>
</tr>
<tr>
<td>TBTS_LEFT</td>
<td>TVM_SETINSERTMARK</td>
</tr>
<tr>
<td>TBTS_RIGHT</td>
<td>TVM_SETINSERTMARKCOLOR</td>
</tr>
<tr>
<td>TBTS_TOP</td>
<td>TVM_SETITEM</td>
</tr>
<tr>
<td>TC_CP_STROKE</td>
<td>TVM_SETITEMA</td>
</tr>
<tr>
<td>TC_CR_90</td>
<td>TVM_SETITEMHEIGHT</td>
</tr>
<tr>
<td>TC_CR_ANY</td>
<td>TVM_SETITEMW</td>
</tr>
<tr>
<td>TC_EA_DOUBLE</td>
<td>TVM_SETLINECOLOR</td>
</tr>
<tr>
<td>TC_IA_ABLE</td>
<td>TVM_SETSCROLLTIME</td>
</tr>
<tr>
<td>TC_OP_CHARACTER</td>
<td>TVM_SETTEXTCOLOR</td>
</tr>
<tr>
<td>TC_OP_STROKE</td>
<td>TVM_SETTOOLTIPS</td>
</tr>
<tr>
<td>TC_RA_ABLE</td>
<td>TVM_SETUNICODEDEFORMATTED</td>
</tr>
<tr>
<td>TC_RESERVED</td>
<td>TVM_SORTCHILDREN</td>
</tr>
<tr>
<td>TC_SA_CONTIN</td>
<td>TVM_SORTCHILDRENCB</td>
</tr>
<tr>
<td>TC_SA_DOUBLE</td>
<td>TVN_BEGINDRAG</td>
</tr>
<tr>
<td>TC_SA_INTEGER</td>
<td>TVN_BEGINDRAGA</td>
</tr>
<tr>
<td>TC_SCROLLBLT</td>
<td>TVN_BEGINDRAGW</td>
</tr>
<tr>
<td>TC_SF_X_YINDEP</td>
<td>TVN_BEGINDLABELEDIT</td>
</tr>
<tr>
<td>TC_SO_ABLE</td>
<td>TVN_BEGINDLABELEDITA</td>
</tr>
<tr>
<td>TC_UA_ABLE</td>
<td>TVN_BEGINDLABELEDITW</td>
</tr>
<tr>
<td>TC_VA_ABLE</td>
<td>TVN_BEGINRDRAG</td>
</tr>
<tr>
<td>TCHT_NOWHERE</td>
<td>TVN_BEGINRDRAGA</td>
</tr>
<tr>
<td>TCHT_ONITEM</td>
<td>TVN_BEGINRDRAGW</td>
</tr>
<tr>
<td>TCHT_ONITEMICON</td>
<td>TVN_DELETEITEM</td>
</tr>
<tr>
<td>TCHT_ONITEMLABEL</td>
<td>TVN_DELETEITEMA</td>
</tr>
<tr>
<td>TCI_SRCCCHARSET</td>
<td>TVN_DELETEITEMW</td>
</tr>
<tr>
<td>TCI_SRCCODEPAGE</td>
<td>TVN_ENDLABELEDIT</td>
</tr>
<tr>
<td>TCI_SRCFONTSIG</td>
<td>TVN_ENDLABELEDITA</td>
</tr>
<tr>
<td>TCI_SRCLOCALE</td>
<td>TVN_ENDLABELEDITW</td>
</tr>
</tbody>
</table>
TCIF_IMAGE
TCIF_PARAM
TCIF_RTLREADING
TCIF_STATE
TCIF_TEXT
TCIS_BUTTONPRESSED
TCIS_HIGHLIGHTED
TCM_ADJUSTRECT
TCM_DELETEALLITEMS
TCM_DELETEITEM
TCM_DESELECTALL
TCM_FIRST
TCM_GETCURFOCUS
TCM_GETCURSEL
TCM_GETEXTENDEDSTYLE
TCM_GETIMAGELIST
TCM_GETITEM
TCM_GETITEMA
TCM_GETITEMCOUNT
TCM_GETITEMRECT
TCM_GETITEMW
TCM_GETROWCOUNT
TCM_GETTOOLTIPS
TCM_GETUNICODEFORMAT
TCM_HIGHLIGHTITEM
TCM_HITTEST
TCM_INSERTITEM
TCM_INSERTITEMA
TCM_INSERTITEMW
TCM_REMOVEIMAGE
TCM_SETCURFOCUS
TCM_SETCURSEL
TCM_SETEXTENDEDSTYLE
TCM_SETIMAGELIST
TCM_SETITEM
TCM_SETITEMA
TCM_SETITEMCOUNT
TCM_SETITEMEXTRA
TCM_SETITEMSIZE
TCM_SETITEMW
TCM_SETMINTABWIDTH
TCM_SETPADDING
TCM_SETTOOLTIPS
TCM_SETUNICODEFORMAT
TCN_FOCUSCHANGE
TCN_GETOBJECT
TVN_GETDISPIINFO
TVN_GETDISPIINFOA
TVN_GETDISPIINFOW
TVN_GETINFOTIP
TVN_GETINFOTIPA
TVN_GETINFOTIPW
TVN_ITEMEXPANDED
TVN_ITEMEXPANDEDA
TVN_ITEMEXPANDEDW
TVN_ITEMEXPANDING
TVN_ITEMEXPANDINGA
TVN_ITEMEXPANDINGW
TVN_KEYDOWN
TVN_SELCHANGED
TVN_SELCHANGEDA
TVN_SELCHANGEDW
TVN_SELCHANGING
TVN_SELCHANGINGA
TVN_SELCHANGINGW
TVN_SETDISPIINFO
TVN_SETDISPIINFOA
TVN_SETDISPIINFOW
TVN_SINGLEXPAND
TVNRET_DEFAULT
TVNRET_SKIPNEW
TVNRET_SKIPOLD
TVS_CHECKBOXES
TVS_DISABLEDRAGDROP
TVS_EDITLABELS
TVS_FULLSCREENSELECT
TVS_HASBUTTONS
TVS_HASLINES
TVS_INFOTIP
TVS_LINESATROOT
TVS_NOHSCROLL
TVS_NOZEROHEIGHT
TVS_NOSCROLL
TVS_NOTOOLTIPS
TVS_RTLREADING
TVS_SINGLEXPAND
TVS_TRACKSELECT
TVSI_NOSINGLEXPAND
TVSIL_NORMAL
TVSIL_STATE
### Constants

**U**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UD_MAXVAL</td>
<td>UDS_HOTTRACK</td>
</tr>
<tr>
<td>UD_MINVAL</td>
<td>UDS_NOTHOUANDS</td>
</tr>
<tr>
<td>UDM_GETACCEL</td>
<td>UDS_SETBUDDYINT</td>
</tr>
<tr>
<td>UDM_GETBASE</td>
<td>UDS_WRAP</td>
</tr>
<tr>
<td>UDM_GETBUDDY</td>
<td>UID_CUT</td>
</tr>
<tr>
<td>UDM_GETPOS</td>
<td>UID_DELETE</td>
</tr>
<tr>
<td>UDM_GETPOS32</td>
<td>UID_DRAGDROP</td>
</tr>
<tr>
<td>UDM_GETRANGE</td>
<td>UID_PASTE</td>
</tr>
<tr>
<td>UDM_GETRANGE32</td>
<td>UID_TYPING</td>
</tr>
<tr>
<td>UDM_GETUNICODEFORMAT</td>
<td>UID_UNKNOWN</td>
</tr>
<tr>
<td>UDM_SETACCEL</td>
<td>UIS_CLEAR</td>
</tr>
<tr>
<td>UDM_SETBASE</td>
<td>UIS_INITIALIZE</td>
</tr>
<tr>
<td>UDM_SETBUDDY</td>
<td>UIS_SET</td>
</tr>
<tr>
<td>UDM_SETPOS</td>
<td>UISF_ACTIVE</td>
</tr>
<tr>
<td>UDM_SETPOS32</td>
<td>UISF_HIDEACCEL</td>
</tr>
<tr>
<td>UDM_SETRANGE</td>
<td>UISF_HIDEFOCUS</td>
</tr>
<tr>
<td>UDM_SETRANGE32</td>
<td>ULW_ALPHA</td>
</tr>
<tr>
<td>UDM_SETUNICODEFORMAT</td>
<td>ULW_COLORKEY</td>
</tr>
<tr>
<td>UDN_DELTAPOS</td>
<td>ULW_OPAQUE</td>
</tr>
<tr>
<td>UDS_ALIGNLEFT</td>
<td>UNICODE_NOCHAR</td>
</tr>
<tr>
<td>UDS_ALIGNRIGHT</td>
<td>UOI_FLAGS</td>
</tr>
<tr>
<td>UDS_ARROWKEYS</td>
<td>UOI_NAME</td>
</tr>
<tr>
<td>UDS_AUTOBUDDY</td>
<td>UOI_TYPE</td>
</tr>
<tr>
<td>UDS_HORZ</td>
<td>UOI_USER_SID</td>
</tr>
</tbody>
</table>

**V**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIABLE_PITCH</td>
<td>VK_MEDIA_NEXT_TRACK</td>
</tr>
<tr>
<td>VERTRES</td>
<td>VK_MEDIA_PLAY_PAUSE</td>
</tr>
<tr>
<td>VERTSIZE</td>
<td>VK_MEDIA_PREV_TRACK</td>
</tr>
<tr>
<td>VIETNAMESE_CHARSET</td>
<td>VK_MEDIA_STOP</td>
</tr>
<tr>
<td>VIEW_DETAILS</td>
<td>VK_MENU</td>
</tr>
<tr>
<td>VIEW_LARGEICONS</td>
<td>VK_MODECHANGE</td>
</tr>
<tr>
<td>VIEW_LIST</td>
<td>VK_MULTIPLY</td>
</tr>
<tr>
<td>VIEW_NETCONNECT</td>
<td>VK_NEXT</td>
</tr>
<tr>
<td>VIEW_NETDISCONNECT</td>
<td>VK_NONAME</td>
</tr>
<tr>
<td>VIEW_NEWFOLDER</td>
<td>VK_NONCONVERT</td>
</tr>
<tr>
<td>VIEW_PARENTFOLDER</td>
<td>VK_NUMLOCK</td>
</tr>
<tr>
<td>VIEW_SMALLICONS</td>
<td>VK_NUMPAD0</td>
</tr>
<tr>
<td>VIEW_SORTDATE</td>
<td>VK_NUMPAD1</td>
</tr>
<tr>
<td>VIEW_SORTNAME</td>
<td>VK_NUMPAD2</td>
</tr>
<tr>
<td>VIEWSORTSIZE</td>
<td>VK_NUMPAD3</td>
</tr>
<tr>
<td>VIEW_SORTTYPE</td>
<td>VK_NUMPAD4</td>
</tr>
<tr>
<td>VIEW_VIEWMENU</td>
<td>VK_NUMPAD5</td>
</tr>
<tr>
<td>Constants</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>VK_ACCEPT</td>
<td>VK_NUMPAD6</td>
</tr>
<tr>
<td>VK_ADD</td>
<td>VK_NUMPAD7</td>
</tr>
<tr>
<td>VK_APPS</td>
<td>VK_NUMPAD8</td>
</tr>
<tr>
<td>VK_ATTN</td>
<td>VK_NUMPAD9</td>
</tr>
<tr>
<td>VK_BACK</td>
<td>VK_OEM_1</td>
</tr>
<tr>
<td>VK_BROWSER_BACK</td>
<td>VK_OEM_102</td>
</tr>
<tr>
<td>VK_BROWSER_FAVORITES</td>
<td>VK_OEM_2</td>
</tr>
<tr>
<td>VK_BROWSER_FORWARD</td>
<td>VK_OEM_3</td>
</tr>
<tr>
<td>VK_BROWSER_HOME</td>
<td>VK_OEM_4</td>
</tr>
<tr>
<td>VK_BROWSER_REFRESH</td>
<td>VK_OEM_5</td>
</tr>
<tr>
<td>VK_BROWSER_SEARCH</td>
<td>VK_OEM_6</td>
</tr>
<tr>
<td>VK_BROWSER_STOP</td>
<td>VK_OEM_7</td>
</tr>
<tr>
<td>VK_CANCEL</td>
<td>VK_OEM_8</td>
</tr>
<tr>
<td>VK_CAPITAL</td>
<td>VK_OEM_ATTN</td>
</tr>
<tr>
<td>VK_CLEAR</td>
<td>VK_OEM_AUTO</td>
</tr>
<tr>
<td>VK_CONTROL</td>
<td>VK_OEM_AX</td>
</tr>
<tr>
<td>VK_CONVERT</td>
<td>VK_OEM_BACKTAB</td>
</tr>
<tr>
<td>VK_CRSSEL</td>
<td>VK_OEM_CLEAR</td>
</tr>
<tr>
<td>VK_DECIMAL</td>
<td>VK_OEM_COMMA</td>
</tr>
<tr>
<td>VK_DELETE</td>
<td>VK_OEM_COPY</td>
</tr>
<tr>
<td>VK_DIVIDE</td>
<td>VK_OEM_CUSEL</td>
</tr>
<tr>
<td>VK_DOWN</td>
<td>VK_OEM_ENLW</td>
</tr>
<tr>
<td>VK_END</td>
<td>VK_OEM_FINISH</td>
</tr>
<tr>
<td>VK_EREOF</td>
<td>VK_OEM_FJ_JISHO</td>
</tr>
<tr>
<td>VK_ESCAPE</td>
<td>VK_OEM_FJ_LOYA</td>
</tr>
<tr>
<td>VK_EXECUTE</td>
<td>VK_OEM_FJ_MASHHOU</td>
</tr>
<tr>
<td>VK_EXSEL</td>
<td>VK_OEM_FJ_ROYA</td>
</tr>
<tr>
<td>VK_F1</td>
<td>VK_OEM_FJ_TOUROKU</td>
</tr>
<tr>
<td>VK_F10</td>
<td>VK_OEM_JUMP</td>
</tr>
<tr>
<td>VK_F11</td>
<td>VK_OEM_MINUS</td>
</tr>
<tr>
<td>VK_F12</td>
<td>VK_OEM_NEC_EQUAL</td>
</tr>
<tr>
<td>VK_F13</td>
<td>VK_OEM_PA1</td>
</tr>
<tr>
<td>VK_F14</td>
<td>VK_OEM_PA2</td>
</tr>
<tr>
<td>VK_F15</td>
<td>VK_OEM_PA3</td>
</tr>
<tr>
<td>VK_F16</td>
<td>VK_OEM_PERIOD</td>
</tr>
<tr>
<td>VK_F17</td>
<td>VK_OEM_PLUS</td>
</tr>
<tr>
<td>VK_F18</td>
<td>VK_OEM_RESET</td>
</tr>
<tr>
<td>VK_F19</td>
<td>VK_OEM_WSCCTRL</td>
</tr>
<tr>
<td>VK_F2</td>
<td>VK_PA1</td>
</tr>
<tr>
<td>VK_F20</td>
<td>VK_PACKET</td>
</tr>
<tr>
<td>VK_F21</td>
<td>VK_PAUSE</td>
</tr>
<tr>
<td>VK_F22</td>
<td>VK_PLAY</td>
</tr>
<tr>
<td>VK_F23</td>
<td>VK_PRINT</td>
</tr>
<tr>
<td>VK_F24</td>
<td>VK_PRIOR</td>
</tr>
<tr>
<td>VK_F3</td>
<td>VK_PROCESSKEY</td>
</tr>
<tr>
<td>VK_F4</td>
<td>VK RBUTTON</td>
</tr>
<tr>
<td>Constant</td>
<td>Constant</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>VK_F5</td>
<td>VK_RCONTROL</td>
</tr>
<tr>
<td>VK_F6</td>
<td>VK_RETURN</td>
</tr>
<tr>
<td>VK_F7</td>
<td>VK_RIGHT</td>
</tr>
<tr>
<td>VK_F8</td>
<td>VK_RMENU</td>
</tr>
<tr>
<td>VK_F9</td>
<td>VK_RSHIFT</td>
</tr>
<tr>
<td>VK_FINAL</td>
<td>VK_RWIN</td>
</tr>
<tr>
<td>VK_HANGEUL</td>
<td>VK_SCROLL</td>
</tr>
<tr>
<td>VK_HANGUL</td>
<td>VK_SELECT</td>
</tr>
<tr>
<td>VK_HANJA</td>
<td>VK_SEPARATOR</td>
</tr>
<tr>
<td>VK_HELP</td>
<td>VK_SHIFT</td>
</tr>
<tr>
<td>VK_HOME</td>
<td>VK_SLEEP</td>
</tr>
<tr>
<td>VK_ICO_00</td>
<td>VK_SNAPSHOT</td>
</tr>
<tr>
<td>VK_ICO_CLEAR</td>
<td>VK_SPACE</td>
</tr>
<tr>
<td>VK_ICO_HELP</td>
<td>VK_SUBTRACT</td>
</tr>
<tr>
<td>VK_INSERT</td>
<td>VK_TAB</td>
</tr>
<tr>
<td>VK_JUNJA</td>
<td>VK_UP</td>
</tr>
<tr>
<td>VK_KANA</td>
<td>VK_VOLUME_DOWN</td>
</tr>
<tr>
<td>VK_KANJI</td>
<td>VK_VOLUME_MUTE</td>
</tr>
<tr>
<td>VK_LAUNCH_APP1</td>
<td>VK_VOLUME_UP</td>
</tr>
<tr>
<td>VK_LAUNCH_APP2</td>
<td>VK_XBUTTON1</td>
</tr>
<tr>
<td>VK_LAUNCH_MAIL</td>
<td>VK_XBUTTON2</td>
</tr>
<tr>
<td>VK_LAUNCH_MEDIA_SELECT</td>
<td>VK_ZOOM</td>
</tr>
<tr>
<td>VK_LBUTTON</td>
<td>VREFRESH</td>
</tr>
<tr>
<td>VK_LCONTROL</td>
<td>VTA_BASELINE</td>
</tr>
<tr>
<td>VK_LEFT</td>
<td>VTA_BOTTOM</td>
</tr>
<tr>
<td>VK_LMENU</td>
<td>VTA_CENTER</td>
</tr>
<tr>
<td>VK_LSHIFT</td>
<td>VTA_LEFT</td>
</tr>
<tr>
<td>VK_LWIN</td>
<td>VTA_RIGHT</td>
</tr>
<tr>
<td>VK_MBUTTON</td>
<td>VTA_TOP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window Procedure</th>
<th>Window Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA_ACTIVE</td>
<td>WM_MENUGETOBJECT</td>
</tr>
<tr>
<td>WA_CLICKACTIVE</td>
<td>WM_MENUBUTTONUP</td>
</tr>
<tr>
<td>WA_INACTIVE</td>
<td>WM_MENUSELECT</td>
</tr>
<tr>
<td>WB_ISDELIMITER</td>
<td>WM_MOUSEACTIVATE</td>
</tr>
<tr>
<td>WB_LEFT</td>
<td>WM_MOUSEFIRST</td>
</tr>
<tr>
<td>WB_RIGHT</td>
<td>WM_MOUSEHOVER</td>
</tr>
<tr>
<td>WGL_FONT_LINES</td>
<td>WM_MOUSELAST</td>
</tr>
<tr>
<td>WGL_FONT_POLYGONS</td>
<td>WM_MOUSELEAVE</td>
</tr>
<tr>
<td>WGL_SWAP_MAIN_PLANE</td>
<td>WM_MOUSEMOVE</td>
</tr>
<tr>
<td>WGL_SWAP_OVERLAY1</td>
<td>WM_MOUSEWHEEL</td>
</tr>
<tr>
<td>WGL_SWAP_OVERLAY10</td>
<td>WM_MOVE</td>
</tr>
<tr>
<td>WGL_SWAP_OVERLAY11</td>
<td>WM_MOVING</td>
</tr>
<tr>
<td>WGL_SWAP_OVERLAY12</td>
<td>WM_NCACTIVATE</td>
</tr>
<tr>
<td>WGL_SWAP_OVERLAY13</td>
<td>WM_NCCALCSIZE</td>
</tr>
</tbody>
</table>
WGL_SWAP_OVERLAY14    WM_NCCREATE
WGL_SWAP_OVERLAY15    WM_NCDestroy
WGL_SWAP_OVERLAY2     WM_NCHittest
WGL_SWAP_OVERLAY3     WM_NCLBUTTONDBLCLK
WGL_SWAP_OVERLAY4     WM_NCLBUTTONDOWN
WGL_SWAP_OVERLAY5     WM_NCLBUTTONUP
WGL_SWAP_OVERLAY6     WM_NCMBUTTONDBLCLK
WGL_SWAP_OVERLAY7     WM_NCMBUTTONDOWN
WGL_SWAP_OVERLAY8     WM_NCMBUTTONUP
WGL_SWAP_OVERLAY9     WM_NCMOUSEHOVER
WGL_SWAP_UNDERLAY1    WM_NCMOUSELEAVE
WGL_SWAP_UNDERLAY10   WM_NCMOUSEMOVE
WGL_SWAP_UNDERLAY11   WM_NCPAINT
WGL_SWAP_UNDERLAY12   WM_NCRBUTTONDBLCLK
WGL_SWAP_UNDERLAY13   WM_NCRBUTTONDOWN
WGL_SWAP_UNDERLAY14   WM_NCRBUTTONUP
WGL_SWAP_UNDERLAY15   WM_NCXBUTTONDBLCLK
WGL_SWAP_UNDERLAY2    WM_NCXBUTTONDOWN
WGL_SWAP_UNDERLAY3    WM_NCXBUTTONUP
WGL_SWAP_UNDERLAY4    WM_NEXTDLGCTL
WGL_SWAP_UNDERLAY5    WM_NEXTMENU
WGL_SWAP_UNDERLAY6    WM_NOTIFY
WGL_SWAP_UNDERLAY7    WM_NOTIFYFORMAT
WGL_SWAP_UNDERLAY8    WM_NULL
WGL_SWAP_UNDERLAY9    WM_PAINT
WGL_SWAPMULTIPLE_MAX WM_PAINTCLIPBOARD
WH_CALLWNDPROC       WM_PAINTICON
WH_CALLWNDPROCRET    WM_PALETTECHANGED
WH_CBT               WM_PALETTEISCHANGING
WH_DEBUG             WM_PARENTNOTIFY
WH_FOREGROUNDIDLE    WM_PASTE
WH_GETMESSAGE       WM_PENWINFIRST
WH_HARDWARE          WM_PENWINLAST
WH_JOURNALPLAYBACK   WM_POWER
WH_JOURNALRECORD     WM_POWERBROADCAST
WH_KEYBOARD          WM_PRINT
WH_KEYBOARD_LL        WM_PRINTCLIENT
WH_MAX               WM_PSD_ENVSTAMPRECT
WH_MAXHOOK           WM_PSD_FULLPAGERECT
WH_MIN               WM_PSD_GREEKTEXTRECT
WH_MINHOOK           WM_PSD_MARGINRECT
WH_MOUSE             WM_PSD_MINMARGINRECT
WH_MOUSE_LL          WM_PSD_PAGESETUPDLG
WH_MSGFILTER         WM_PSD_YAFULLPAGERECT
WH_SHELL             WM_QUERYDRAGICON
WH_SYSMSGFILTER      WM_QUERYENDSESSION
<table>
<thead>
<tr>
<th>Constant</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEEL_DELTA</td>
<td>WM_QUERYNEWPALETTE</td>
</tr>
<tr>
<td>WHITE_BRUSH</td>
<td>WM_QUERYPEN</td>
</tr>
<tr>
<td>WHITE_PEN</td>
<td>WM_QUERYUISTATE</td>
</tr>
<tr>
<td>WHITEONBLACK</td>
<td>WM_QUEUESYNC</td>
</tr>
<tr>
<td>WINDING</td>
<td>WM_QUIT</td>
</tr>
<tr>
<td>WINEVENT_INCONTEXT</td>
<td>WM_RBUTTONDBLCLK</td>
</tr>
<tr>
<td>WINEVENT_OUTOFCONTEXT</td>
<td>WM_RBUTTONDOWN</td>
</tr>
<tr>
<td>WINEVENT_SKIPOWNPROCESS</td>
<td>WM_RBUTTONUP</td>
</tr>
<tr>
<td>WINEVENT_SKIPOWNTHREAD</td>
<td>WM_RENDERALLFORMATS</td>
</tr>
<tr>
<td>WINSTA_ACCESSCLIPBOARD</td>
<td>WM_RENDERFORMAT</td>
</tr>
<tr>
<td>WINSTA_ACCESSGLOBALATOMS</td>
<td>WM_SET.minLength</td>
</tr>
<tr>
<td>WINSTA_ALL_ACCESS</td>
<td>WM_SET_FOCUS</td>
</tr>
<tr>
<td>WINSTA_CREATEDESKTOP</td>
<td>WM_SETFont</td>
</tr>
<tr>
<td>WINSTA_ENUMDESKTOPS</td>
<td>WM_SETHOTKEY</td>
</tr>
<tr>
<td>WINSTA_ENUMERATE</td>
<td>WM_SETICON</td>
</tr>
<tr>
<td>WINSTA_EXITWINDOWS</td>
<td>WM_SETREDRAW</td>
</tr>
<tr>
<td>WINSTA_READATTRIBUTES</td>
<td>WM_SETTEXT</td>
</tr>
<tr>
<td>WINSTA_READSCREEN</td>
<td>WM_SETSETTINGCHANGE</td>
</tr>
<tr>
<td>WINSTA_WRITEATTRIBUTES</td>
<td>WM_SHOWWINDOW</td>
</tr>
<tr>
<td>WINVER</td>
<td>WM_SIZE</td>
</tr>
<tr>
<td>WM_ACTIVATE</td>
<td>WM_SIZECLIPBOARD</td>
</tr>
<tr>
<td>WM_ACTIVATEAPP</td>
<td>WM_SIZING</td>
</tr>
<tr>
<td>WM_AFXFIRST</td>
<td>WM_SPOOLERSTATUS</td>
</tr>
<tr>
<td>WM_AFXLAST</td>
<td>WM_STYLECHANGED</td>
</tr>
<tr>
<td>WM_APP</td>
<td>WM_STYLECHANGING</td>
</tr>
<tr>
<td>WM_APPCOMMAND</td>
<td>WM_SYNCPAINT</td>
</tr>
<tr>
<td>WM_ASKCBFORMATNAME</td>
<td>WM_SYSCCHAR</td>
</tr>
<tr>
<td>WM.CancelJournal</td>
<td>WM_SYSCOLORCHANGE</td>
</tr>
<tr>
<td>WM_CANCELMODE</td>
<td>WM_SYSCOMMAND</td>
</tr>
<tr>
<td>WM_CAPTURECHANGED</td>
<td>WM_SYSDEADCHAR</td>
</tr>
<tr>
<td>WM_CHANGECBOCHAIN</td>
<td>WM_SYSKEYDOWN</td>
</tr>
<tr>
<td>WM_CHANGEUISTATE</td>
<td>WM_SYSKEYUP</td>
</tr>
<tr>
<td>WM_CHAR</td>
<td>WM_TABLET_FIRST</td>
</tr>
<tr>
<td>WM_CHARTOITEM</td>
<td>WM_TABLET_LAST</td>
</tr>
<tr>
<td>WM_CHILDACTIVATE</td>
<td>WM_TCARD</td>
</tr>
<tr>
<td>WM_CHOOSEFONT_GETLOGFONT</td>
<td>WM_THEMENCHANGED</td>
</tr>
<tr>
<td>WM_CHOOSEFONT_SETFLAGS</td>
<td>WM_TIMECHANGE</td>
</tr>
<tr>
<td>WM_CHOOSEFONT_SETLOGFONT</td>
<td>WM_TIMER</td>
</tr>
<tr>
<td>WM_CLEAR</td>
<td>WM_UNDO</td>
</tr>
<tr>
<td>WM_CLOSE</td>
<td>WM_UNICHAR</td>
</tr>
<tr>
<td>WM_COMMAND</td>
<td>WM_UNINITMENUPOPUP</td>
</tr>
<tr>
<td>WM_COMMNOTIFY</td>
<td>WM_UPDATEUISTATE</td>
</tr>
<tr>
<td>WM_COMPACTING</td>
<td>WM_USER</td>
</tr>
<tr>
<td>WM_COMPAREITEM</td>
<td>WM_USERCHANGED</td>
</tr>
<tr>
<td>WM_CONTEXTMENU</td>
<td>WM_VKEYTOITEM</td>
</tr>
<tr>
<td>WM_COPY</td>
<td>WM_VSCROLL</td>
</tr>
<tr>
<td>Constant</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>WM_COPYDATA</td>
<td>WM_COPYDATA</td>
</tr>
<tr>
<td>WM_CREATE</td>
<td>WM_CREATE</td>
</tr>
<tr>
<td>WM_CTLCOLORBTN</td>
<td>WM_CTLCOLORBTN</td>
</tr>
<tr>
<td>WM_CTLCOLORDLG</td>
<td>WM_CTLCOLORDLG</td>
</tr>
<tr>
<td>WM_CTLCOLOREDIT</td>
<td>WM_CTLCOLOREDIT</td>
</tr>
<tr>
<td>WM_CTLCOLORLISTBOX</td>
<td>WM_CTLCOLORLISTBOX</td>
</tr>
<tr>
<td>WM_CTLCOLORMSGBOX</td>
<td>WM_CTLCOLORMSGBOX</td>
</tr>
<tr>
<td>WM_CTLCOLORSCROLLBAR</td>
<td>WM_CTLCOLORSCROLLBAR</td>
</tr>
<tr>
<td>WM_CTLCOLORSTATIC</td>
<td>WM_CTLCOLORSTATIC</td>
</tr>
<tr>
<td>WM_DEADCHAR</td>
<td>WM_DEADCHAR</td>
</tr>
<tr>
<td>WM_DELETEITEM</td>
<td>WM_DELETEITEM</td>
</tr>
<tr>
<td>WM_DESTROY</td>
<td>WM_DESTROY</td>
</tr>
<tr>
<td>WM_DESTROYCLIPBOARD</td>
<td>WM_DESTROYCLIPBOARD</td>
</tr>
<tr>
<td>WM_DEVICECHANGE</td>
<td>WM_DEVICECHANGE</td>
</tr>
<tr>
<td>WM_DEVMODECHANGE</td>
<td>WM_DEVMODECHANGE</td>
</tr>
<tr>
<td>WM_DISPLAYCHANGE</td>
<td>WM_DISPLAYCHANGE</td>
</tr>
<tr>
<td>WM_DRAWCLIPBOARD</td>
<td>WM_DRAWCLIPBOARD</td>
</tr>
<tr>
<td>WM_DRAWITEM</td>
<td>WM_DRAWITEM</td>
</tr>
<tr>
<td>WM_DROPFILES</td>
<td>WM_DROPFILES</td>
</tr>
<tr>
<td>WM_ENABLE</td>
<td>WM_ENABLE</td>
</tr>
<tr>
<td>WM_ENDSESSION</td>
<td>WM_ENDSESSION</td>
</tr>
<tr>
<td>WM_ENDSESSION</td>
<td>WM_ENDSESSION</td>
</tr>
<tr>
<td>WM_ENTERIDLE</td>
<td>WM_ENTERIDLE</td>
</tr>
<tr>
<td>WM_ENTERMENULOOP</td>
<td>WM_ENTERMENULOOP</td>
</tr>
<tr>
<td>WM_ENTERMENULOOP</td>
<td>WM_ENTERMENULOOP</td>
</tr>
<tr>
<td>WM_ENTERSIZEMOVE</td>
<td>WM_ENTERSIZEMOVE</td>
</tr>
<tr>
<td>WM_ERASEBKGD</td>
<td>WM_ERASEBKGD</td>
</tr>
<tr>
<td>WM_EXITMENULOOP</td>
<td>WM_EXITMENULOOP</td>
</tr>
<tr>
<td>WM_EXITSIZEMOVE</td>
<td>WM_EXITSIZEMOVE</td>
</tr>
<tr>
<td>WM_FONTCHANGE</td>
<td>WM_FONTCHANGE</td>
</tr>
<tr>
<td>WM_GETDLGCODE</td>
<td>WM_GETDLGCODE</td>
</tr>
<tr>
<td>WM_GETFONT</td>
<td>WM_GETFONT</td>
</tr>
<tr>
<td>WM_GETHOTKEY</td>
<td>WM_GETHOTKEY</td>
</tr>
<tr>
<td>WM_GETICON</td>
<td>WM_GETICON</td>
</tr>
<tr>
<td>WM_GETMINMAXINFO</td>
<td>WM_GETMINMAXINFO</td>
</tr>
<tr>
<td>WM.GetObject</td>
<td>WM.GetObject</td>
</tr>
<tr>
<td>WM_GETTEXT</td>
<td>WM_GETTEXT</td>
</tr>
<tr>
<td>WM_GETTEXTLENGTH</td>
<td>WM_GETTEXTLENGTH</td>
</tr>
<tr>
<td>WM_HANDHELDFIRST</td>
<td>WM_HANDHELDFIRST</td>
</tr>
<tr>
<td>WM_HANDBLEDFIRST</td>
<td>WM_HANDBLEDFIRST</td>
</tr>
<tr>
<td>WM_HOTKEY</td>
<td>WM_HOTKEY</td>
</tr>
<tr>
<td>WM_HSCROLL</td>
<td>WM_HSCROLL</td>
</tr>
<tr>
<td>WM_HSCROLLCLIPBOARD</td>
<td>WM_HSCROLLCLIPBOARD</td>
</tr>
<tr>
<td>WM_ICONERASEBKGD</td>
<td>WM_ICONERASEBKGD</td>
</tr>
<tr>
<td>WM_IME_CHAR</td>
<td>WM_IME_CHAR</td>
</tr>
<tr>
<td>WM_IME_COMPOSITION</td>
<td>WM_IME_COMPOSITION</td>
</tr>
<tr>
<td>WM_WTSSESSION_CHANGE</td>
<td>WM_WTSSESSION_CHANGE</td>
</tr>
<tr>
<td>WM_WINDOWPOSCHANGED</td>
<td>WM_WINDOWPOSCHANGED</td>
</tr>
<tr>
<td>WM_WINDOWPOSCHANGING</td>
<td>WM_WINDOWPOSCHANGING</td>
</tr>
<tr>
<td>WM_WININICHANGE</td>
<td>WM_WININICHANGE</td>
</tr>
<tr>
<td>WM_XBUTTONDOWN</td>
<td>WM_XBUTTONDOWN</td>
</tr>
<tr>
<td>WM_XBUTTONDBLCLK</td>
<td>WM_XBUTTONDBLCLK</td>
</tr>
<tr>
<td>WM_XBUTTONDOWN</td>
<td>WM_XBUTTONDOWN</td>
</tr>
<tr>
<td>WMSZ_BOTTOM</td>
<td>WMSZ_BOTTOM</td>
</tr>
<tr>
<td>WMSZ_BOTTOMLEFT</td>
<td>WMSZ_BOTTOMLEFT</td>
</tr>
<tr>
<td>WMSZ_BOTTOMRIGHT</td>
<td>WMSZ_BOTTOMRIGHT</td>
</tr>
<tr>
<td>WMSZ_LEFT</td>
<td>WMSZ_LEFT</td>
</tr>
<tr>
<td>WMSZ_RIGHT</td>
<td>WMSZ_RIGHT</td>
</tr>
<tr>
<td>WMSZ_TOP</td>
<td>WMSZ_TOP</td>
</tr>
<tr>
<td>WMSZ_TOPLEFT</td>
<td>WMSZ_TOPLEFT</td>
</tr>
<tr>
<td>WMSZ_TOPRIGHT</td>
<td>WMSZ_TOPRIGHT</td>
</tr>
<tr>
<td>WPF_ASYNCWINDOWPLACEMENT</td>
<td>WPF_ASYNCWINDOWPLACEMENT</td>
</tr>
<tr>
<td>WPF_RESTORETOMAXIMIZED</td>
<td>WPF_RESTORETOMAXIMIZED</td>
</tr>
<tr>
<td>WPF_SETIMPOSITION</td>
<td>WPF_SETIMPOSITION</td>
</tr>
<tr>
<td>WS_ACTIVECAPTION</td>
<td>WS_ACTIVECAPTION</td>
</tr>
<tr>
<td>WS_BORDER</td>
<td>WS_BORDER</td>
</tr>
<tr>
<td>WS_CAPTION</td>
<td>WS_CAPTION</td>
</tr>
<tr>
<td>WS_CHILD</td>
<td>WS_CHILD</td>
</tr>
<tr>
<td>WS_CHILDWINDOW</td>
<td>WS_CHILDWINDOW</td>
</tr>
<tr>
<td>WS_CLIPCHILDREN</td>
<td>WS_CLIPCHILDREN</td>
</tr>
<tr>
<td>WS_CLIPSIBLINGS</td>
<td>WS_CLIPSIBLINGS</td>
</tr>
<tr>
<td>WS_DISABLED</td>
<td>WS_DISABLED</td>
</tr>
<tr>
<td>WS_DLGMODALFRAME</td>
<td>WS_DLGMODALFRAME</td>
</tr>
<tr>
<td>WS_EX_ACCEPTFILES</td>
<td>WS_EX_ACCEPTFILES</td>
</tr>
<tr>
<td>WS_EX_APPWINDOW</td>
<td>WS_EX_APPWINDOW</td>
</tr>
<tr>
<td>WS_EX_CLIENTEDGE</td>
<td>WS_EX_CLIENTEDGE</td>
</tr>
<tr>
<td>WS_EX_COMPOSITED</td>
<td>WS_EX_COMPOSITED</td>
</tr>
<tr>
<td>WS_EX_CONTEXTHELP</td>
<td>WS_EX_CONTEXTHELP</td>
</tr>
<tr>
<td>WS_EX_CONTROLPARENT</td>
<td>WS_EX_CONTROLPARENT</td>
</tr>
<tr>
<td>WS_EX_DLGMODALFRAME</td>
<td>WS_EX_DLGMODALFRAME</td>
</tr>
<tr>
<td>WS_EX_LAYERED</td>
<td>WS_EX_LAYERED</td>
</tr>
<tr>
<td>WS_EX_LAYOUTRTL</td>
<td>WS_EX_LAYOUTRTL</td>
</tr>
<tr>
<td>WS_EX_LEFT</td>
<td>WS_EX_LEFT</td>
</tr>
<tr>
<td>WS_EX_LEFTSCROLLBAR</td>
<td>WS_EX_LEFTSCROLLBAR</td>
</tr>
<tr>
<td>WS_EX_LTRREADING</td>
<td>WS_EX_LTRREADING</td>
</tr>
<tr>
<td>WS_EX_MDICHILD</td>
<td>WS_EX_MDICHILD</td>
</tr>
<tr>
<td>WS_EX_NOACTIVATE</td>
<td>WS_EX_NOACTIVATE</td>
</tr>
<tr>
<td>WS_EX_NOMINHERITLAYOUT</td>
<td>WS_EX_NOMINHERITLAYOUT</td>
</tr>
<tr>
<td>WS_EX_NOPARENTNOTIFY</td>
<td>WS_EX_NOPARENTNOTIFY</td>
</tr>
<tr>
<td>WS_EX_OVERLAPPEDWINDOW</td>
<td>WS_EX_OVERLAPPEDWINDOW</td>
</tr>
<tr>
<td>Constant</td>
<td>Style</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>WM_IME_COMPOSITIONFULL</td>
<td>WS_EX_RIGHT</td>
</tr>
<tr>
<td>WM_IME_CONTROL</td>
<td>WS_EX_RIGHTSCROLLBAR</td>
</tr>
<tr>
<td>WM_IME_ENDCOMPOSITION</td>
<td>WS_EX_RTLREADING</td>
</tr>
<tr>
<td>WM_IME_KEYDOWN</td>
<td>WS_EX_STATICEDGE</td>
</tr>
<tr>
<td>WM_IME_KEYLAST</td>
<td>WS_EX_TOOLWINDOW</td>
</tr>
<tr>
<td>WM_IME_KEYUP</td>
<td>WS_EX_TOPMOST</td>
</tr>
<tr>
<td>WM_IME_NOTIFY</td>
<td>WS_EX_TRANSPARENT</td>
</tr>
<tr>
<td>WM_IME_REQUEST</td>
<td>WS_EX_WINDOWEDGE</td>
</tr>
<tr>
<td>WM_IME_SELECT</td>
<td>WS_GROUP</td>
</tr>
<tr>
<td>WM_IME_SETCONTEXT</td>
<td>WS_HSCROLL</td>
</tr>
<tr>
<td>WM_IME_STARTCOMPOSITION</td>
<td>WS_ICONIC</td>
</tr>
<tr>
<td>WM_INITDIALOG</td>
<td>WS_MAXIMIZE</td>
</tr>
<tr>
<td>WM_INITMENU</td>
<td>WS_MAXIMIZEBOX</td>
</tr>
<tr>
<td>WM_INITMENUPopup</td>
<td>WS_MINIMIZE</td>
</tr>
<tr>
<td>WM_INPUT</td>
<td>WS_MINIMIZEBOX</td>
</tr>
<tr>
<td>WM_INPUTLANGCHANGE</td>
<td>WS_OVERLAPPED</td>
</tr>
<tr>
<td>WM_INPUTLANGCHANGEREQUEST</td>
<td>WS_OVERLAPPEDWINDOW</td>
</tr>
<tr>
<td>WM_KEYDOWN</td>
<td>WS_POPUP</td>
</tr>
<tr>
<td>WM_KEYFIRST</td>
<td>WS_POPUPWINDOW</td>
</tr>
<tr>
<td>WM_KEYLAST</td>
<td>WS_SIZEBOX</td>
</tr>
<tr>
<td>WM_KEYUP</td>
<td>WS_SYSMENU</td>
</tr>
<tr>
<td>WM_KILLFOCUS</td>
<td>WS_TABSTOP</td>
</tr>
<tr>
<td>WM_LBUTTONDBLCLK</td>
<td>WS_THICKFRAME</td>
</tr>
<tr>
<td>WM_LBUTTONDOWN</td>
<td>WS_TILED</td>
</tr>
<tr>
<td>WM_LBUTTONUP</td>
<td>WS_VISIBLE</td>
</tr>
<tr>
<td>WM_MBUTTONDBLCLK</td>
<td>WS_VSCROLL</td>
</tr>
<tr>
<td>WM_MBUTTONDOWN</td>
<td>WS_VISIBLE</td>
</tr>
<tr>
<td>WM_MBUTTONDOWN</td>
<td>WTS_CONSOLE_CONNECT</td>
</tr>
<tr>
<td>WM_MDIACTIVATE</td>
<td>WTS_CONSOLE_DISCONNECT</td>
</tr>
<tr>
<td>WM_MDICASCADE</td>
<td>WTS_REMOTE_CONNECT</td>
</tr>
<tr>
<td>WM_MDICREATE</td>
<td>WTS_REMOTE_DISCONNECT</td>
</tr>
<tr>
<td>WM_MDIDESTROY</td>
<td>WTS_SESSION_LOCK</td>
</tr>
<tr>
<td>WM_MDIGETACTIVE</td>
<td>WTS_SESSION_LOGOFF</td>
</tr>
<tr>
<td>WM_MDIICONARRANGE</td>
<td>WTS_SESSION_LOGON</td>
</tr>
<tr>
<td>WM_MDIMAXIMIZE</td>
<td>WTS_SESSION_REMOTE_CONTROL</td>
</tr>
<tr>
<td>WM_MDINEXT</td>
<td>WTS_SESSION_UNLOCK</td>
</tr>
<tr>
<td>WM_MDIREFRESHMENU</td>
<td>WVR_ALIGNBOTTOM</td>
</tr>
<tr>
<td>WM_MDIREDRAW</td>
<td>WVR_ALIGNLEFT</td>
</tr>
<tr>
<td>WM_MDISETMENU</td>
<td>WVR_ALIGNRIGHT</td>
</tr>
<tr>
<td>WM_MDISETMENU</td>
<td>WVR_ALIGNRIGHT</td>
</tr>
<tr>
<td>WM_MDITILE</td>
<td>WVR_ALIGNTOP</td>
</tr>
<tr>
<td>WM_MEASUREITEM</td>
<td>WVR_HREDRAW</td>
</tr>
<tr>
<td>WM_MENUCHAR</td>
<td>WVR_REDRAW</td>
</tr>
<tr>
<td>WM_MENUCOMMAND</td>
<td>WVR_VALIDRECTS</td>
</tr>
<tr>
<td>WM_MENUDRAW</td>
<td>WVR_VREDRAW</td>
</tr>
</tbody>
</table>
Vine™ Add-in
# Table of Contents

User’s Guide .................................................................................................................... 1
Getting Started ...................................................................................................... 1
Overview ..................................................................................................... 1
Installation .................................................................................................. 1
Demo Workbook ......................................................................................... 1
Video ........................................................................................................... 1
Making a Connection ............................................................................................ 1
Adding Data Points ............................................................................................... 3
Working with Ranges .......................................................................................... 5
Demo .................................................................................................................. 5
Rotate or Cycle ............................................................................................ 8
Troubleshooting .................................................................................................... 9
Reference ...................................................................................................................... 11
Connection .......................................................................................................... 11
Data Points ......................................................................................................... 13
Log .......................................................................................................................... 15
User's Guide

Getting Started

Overview

Skkynet Vine Add-in for Microsoft Excel connects Excel 2010, 2013, 2016 or 2019 to Vine Server, Cogent DataHub, or SkkyHub. This manual assumes that you have an installed Vine Server or Cogent DataHub, or that you have access to a SkkyHub account.

Installation

To install Vine Add-in simply download the program archive, double-click on it, and follow the instructions. You will need to close Excel for the installation.

Demo Workbook

The Vine Add-in installation includes a demo workbook that provides a step-by-step, hands-on demonstration of the most-used features. We encourage first-time users to go through that first, and refer to this documentation as a secondary resource.

Video

Here is a video that shows how to connect Vine Add-in to the DataHub.

Making a Connection

Once Vine Add-in is installed, you are ready to make a connection.

1. In Excel, click on the Skkynet Vine tab, and then choose Open Configuration from the ribbon menu.

2. In the Connection tab you can configure a connection to Vine Server, Cogent DataHub, or SkkyHub.
3. Enter the **Host Name** or IP address and the **Port** number for the host computer for your Vine Server or DataHub, or the IP address for your SkkyHub service. If you will be making a WebSocket connection (see below), the port number is typically 80, and if you are using SSL (with or without WebSockets), it's typically 443. Connections to SkkyHub must use these port numbers. They must also use WebSocket, and SSL is recommended.

4. Enter your **User Name** and **Password** for Vine Server or DataHub (if any) or SkkyHub. You can check the **Save** box to save your password with this worksheet.

5. In the **Options**, you can select **WebSocket** and/or **SSL** as appropriate. The **Accept invalid certificates** option is there in case your certificate has expired.

6. Checking the **Connect automatically** box tells Vine Add-in to make the connection to the data source as soon as the worksheet is opened.

7. You can configure **Throttle updates** to control the rate (in milliseconds) at which updates come into Excel over this connection. This is helpful for reducing bandwidth or system resource use. The default (0) allows updates as fast as possible.

8. You will need to configure **Proxy Settings** if there is a proxy between you and your Vine Server, DataHub or SkkyHub. Enter the **Host Name**, **Port**, **User Name** and **Password** for the proxy here.

9. The **Security Settings** are not yet documented.
10. When you are ready to connect, click the Connect button. The status should eventually change to Connection succeeded: Connected.

If the connect attempt fails, you can check the Log to see what the connection steps were, and get an indication of what might have gone wrong.

Once your connection is established, you are ready to add some data points.

Adding Data Points

When you are connected to Vine Server, DataHub or SkkyHub, you can start adding data points.

1. In the Data Points tab, select a point from the Points list in the lower frame.

2. Notice that Vine Add-in automatically fills in the Selected Point and Range fields, and automatically generates a Range Name. You can choose a different cell to change the range, and you can edit the range name, if desired.
3. When you are ready to add the point, click the **Add** button...

...and the point will be added.

At any time you can change the entered parameters for an existing point, and then click the **Update** button to have your changes take effect.

4. You can also add information about a point other than its value, using the Properties drop-down box. For example, go to another cell and then select **Quality Name** for the same point from the Properties list.
5. Click the **Add** button, and the quality name will get entered.

6. To add a point that writes back to Vine Server or SkkyHub, you can use the **Direction** option to change the direction to **Read** or **Write** for one-way data flow.

![Image showing the Add button and bindings tab]

For bidirectional data flow, **Read before Write** always takes the value from Vine Server, DataHub or SkkyHub initially when this worksheet is first opened. **Write before Read**, on the other hand, updates Vine Server, DataHub or SkkyHub as an initial value. In either case, once the connection is made, the data is subsequentially updated by the latest values from either side.

7. To see a list of the points you have added, click the **Bindings** tab.

![Image showing the Bindings tab]

Here you can enable or disable any data point bindings you have made, or remove them.

Now that you are familiar with adding individual points, you are ready to **work with ranges**. For more details adding points, please refer to the section called “Data Points”.

### Working with Ranges

#### Demo

In addition to single cells, you can bind a range of cells to a single Vine Server, DataHub or SkkyHub point. Here is a brief demo of how to work with ranges:

1. Select a range.
2. In the **Data Points** tab, select or enter a Vine Server, DataHub or SkkyHub data point name. For the purposes of this demo, this should be a data point that you can write to.

3. In the **Direction** option, choose **Write**.

4. Click the **Add** button to create the binding. This will act as your data source.

5. If you have access to your Vine Server, DataHub or SkkyHub data browser, you can check to see that the point exists, that it is bound, and that it is a range.

6. Now you can create a second range to receive the data. Ensure that you still have the same **Selected Point**, and then in the worksheet select a single cell.

7. Click the the range size icon. The one-cell selection should expand to the size of the bound range.
8. In the **Direction** option, choose **Read**.

9. Click the **Add** button.

![Image of a spreadsheet with cells and buttons]

The range should fill with the data from the data source range. Now, whenever any value in data source range changes, this range will update with the same values.

10. What happens if the size of the source range gets changed? You can configure this target range to automatically change as well. While the target range is still selected, check the **Auto** box, and then click **Update**.

![Image of a spreadsheet with checkmark]

11. Now change the size of the source range, say by adding an extra row. First, select the source range.

![Image of a spreadsheet with selected rows and a dialog box]

An easy way to select a range is from the **Bindings** tab.

12. With the source range selected, click the new range icon.

13. Enter the coordinates for the new range, click **OK**.

14. Click **Update**.
The target range should update as well.

**Rotate or Cycle**

A multi-cell range can be used to create a rotating or cycling series of updates through the range. The range must be either a single row or a single column, and it must be bound to a point with a single value.

For example, using the DataSim program, you can bind a vertical range to the **DataSim::Ramp** point. As the point values increase, the latest values appear at the bottom of the column, and each older value is pushed up one cell. You can check the Fill Style's **Reverse** box to change directions, and have the latest values appear at the top.
The Fill Style options let you specify Vertical or Horizontal directions for columns or rows, respectively. Also, you can choose between Rotate and Cycle. The Cycle option cycles the latest value through the different positions in the range, creating a wave of updates. Again, checking the Reverse box changes the direction.

Troubleshooting

Protected View messages

Under certain circumstances, such as when you download a workbook from the Internet, or copy it from another computer on your network, you may see the following message when trying to open the file. This will prevent your data from updating.

Clicking the Enable Editing button will close the message, but your data may still not update. You will need to close and reopen the file in order for the data to start updating.

If you still can’t see the date, then you may need to close the workbook and open its Properties window in the file browser. At the bottom of the General tab, check to see if there is a Security option, with an Unblock box checked.
If that box is checked, uncheck it, and then click **OK**. When you reopen the workbook, your data should start updating.

**What to do if you can't connect**

1. Open the Vine configuration panel and look in the **Log** tab. If you see messages that look like this:

<table>
<thead>
<tr>
<th>Connection failed - retrying: One or more errors occurred.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to connect to the remote server:</td>
</tr>
<tr>
<td>The remote name could not be resolved: 'demo.skkyenet.com'</td>
</tr>
</tbody>
</table>

   This means your computer could not connect to the cloud server. Typically, if you can connect to a web site from your computer, then you should be able to reach the demo.skkyenet.com demo site.

2. The SkkynetVineDemo.xlsx workbook tries to make a WebSocket connection to the SkkyHub cloud server to read and write data. If you are using Windows 7 you will find the connection will not succeed because the WebSocket implementation used by Vine Add-in is not supported in this OS. Please try the demo on a Windows 8 or later system, or you can connect to a local copy of the DataHub or Vine Server to stream data across your network.

**Other information**

- Streaming live data disables Excel's Undo feature. To enable the Undo feature again, you will need to disconnect from the data stream.

**Contact Support**

If none of the above suggestions solve your problem, you can send email to support@skkyenet.com and we will respond as soon as possible. Go to Options
Reference

Connection

These are the required and optional parameters for making a connection to Vine Server, DataHub or SkkyHub.

Host Name
The name or IP address of the host computer for Vine Server or DataHub, or the IP address for the SkkyHub service.

Port
The port number on the host computer or SkkyHub service. Port 80 is standard for WebSocket connections, and 443 for SSL (see below).

User Name
The user name required by SkkyHub, DataHub or Vine Server, if any.

Password
The corresponding password. The **Save** button allows you to save the password with this worksheet.

Options

**Use WebSocket**
This is required for connections to SkkyHub, and optional for DataHub or Vine Server. It ensures that the connection is outbound-only from Vine Add-in. Port 80 (see above) is standard for this option when used without SSL.

**Use SSL**
Provides the option of using SSL. Port 443 (see above) is standard for this option, when used alone or with the WebSocket option.

**Accept invalid certificates**
 Allows the SSL connection to be made even if the security certificate is invalid.
**Connect automatically**
Attempts to make the connection to the data source when the worksheet is opened.

**Throttle updates**
Lets you control the rate (in milliseconds) at which updates come into Excel over this connection. This is helpful for reducing bandwidth or system resource use. The default (0) allows updates as fast as possible.

**Proxy Settings**

![Proxy Settings](image)

**Host Name**
The name or IP address of the host computer for the proxy server.

**Port**
The port number on the host computer for the proxy server.

**User Name**
The user name required by the proxy server, if any.

**Password**
The corresponding password.

**Security Settings**

![Security Settings](image)

**Config Password**
Entering a password here and saving the worksheet will encrypt the Vine Add-in configuration stored within the worksheet. Once a password is set, any user opening the worksheet would need to open the Skkynet Vine panel and enter the password to establish the data connection.

**For all connections**

![Connect Button](image)

**Connect button**
Sends a request for a connection (or disconnect) according to the current configuration.
tion.

Status messages
Displays the status of the connection. These all get logged, and can be viewed in the Log tab.

Data Points
Here you can select and configure the data points to which you want to connect.

Selected Point
Enter the point name manually, or select it from the list in Points or Bindings below.

Range
The range selected in the worksheet will appear here. You can change this with the new range icon or modify it to match the size of the range with the range size icon. You can also check the Auto box to automatically adjust the range when the range in the data point changes size. These changes will take effect when you click the Update button (below).

Range Name
The system provides a default range name, which you can change. This range name can be used in formulas in other cells.

Options

Direction
There are four options for the direction of the data flow:

- **Read** the data from the server into Excel.
- **Write** the data from Excel to the server. Use this to publish a value or formula result to the server.
- **Read before Write** creates a bidirectional connection that always takes the value from the server initially, when this worksheet is first opened. Once the connection is
made, the data is sequentially updated by the latest values from either side.

- **Write before Read** creates a bidirectional connection that always *updates the server* from this worksheet when it is first opened. Once the connection is made, the data is sequentially updated by the latest values from either side.

**Properties**

Lets you select which properties of the data will appear in consecutive cells in a range (**UTC Time**, **Local Time**, **Value**, **Quality**, and **Quality Name**). They will be added to the range in that order. You need to ensure that you have selected the correct number of cells to accommodate all of the ones you select.

**Treatment**

Lets you specify how the data is treated. **Value** preserves the point's data type and value. **Formula** treats strings that start with `=` as formulas; all others are treated the same as **Value**.

**Fill Style**

When you select a range of cells, the **Fill Style** determines how **Read** values will fill the cell:

- **Fill** updates all cells in the range with the same value.
- **RotateVertical** adds new values to the last cell in the range, pushing all previous values upward within the range.
- **RotateHorizontal** adds new values to the last cell in the range, pushing all previous values to the left.
- **CycleVertical** adds new values to the range from top to bottom. When the last cell in the range is filled, new values begin to to fill in from the top again.
- **CycleHorizontal** adds new values to the range from left to right. When the last cell in the range is filled, new values begin to to fill in from the left again.
- **Reverse** changes the direction of any **Rotate** or **Cycle** option above (e.g. bottom to top, right to left, etc.).

**For all selections**

- **Add** button
  - Adds the current selection.
- **Update** button
  - Applies any configuration changes made to the current selection.
- **Points**
  - Displays all the points that are available on the server.
**Bindings**
Displays all the saved bindings of points to ranges.

**Remove Selected** button
Removes the selected binding, leaving the data in the cell.

**Remove and Clear Selected** button
Removes the selected binding and clears the cell.

**Log**
Each status message for the connection is logged, and can be viewed here.
Cogent DataHub® APIs for C++, Java, and .NET
Version 9.0
# Table of Contents

Introduction .................................................................................................................... 1  
  Preliminaries ................................................................................................................ 1  
  C++ Programming ........................................................................................................ 2  
  Java Programming ....................................................................................................... 2  
  .NET Programming ...................................................................................................... 4  
  Example Programs ......................................................................................................... 4  

The DataHubConnector Class ......................................................................................... 5  
  Overview ....................................................................................................................... 5  
  Categorized List of Methods ...................................................................................... 5  
  Making Callbacks ......................................................................................................... 7  

The DataHubPoint Class ................................................................................................. 8  
  Overview ....................................................................................................................... 8  
  Categorized List of Methods ...................................................................................... 9  

DHTP - The DataHub Transfer Protocol ....................................................................... 11  
  DHTP - The DataHub Transfer Protocol .................................................................... 11  

A. GNU General Public License ................................................................................... 13  
B. GNU Lesser General Public License ........................................................................... 20  

I. DataHubConnector Methods ..................................................................................... 30  
  DataHubConnector ...................................................................................................... 31  
  ~DataHubConnector .................................................................................................... 32  
  activeHeartbeatTimers .............................................................................................. 33  
  addPointValue ............................................................................................................ 35  
  appendPointValue ...................................................................................................... 37  
  cancelHeartbeatTimers .............................................................................................. 39  
  cancelReconnectionTimer .......................................................................................... 40  
  closeConnection ......................................................................................................... 41  
  createPoint ................................................................................................................ 42  
  dividePointValue ...................................................................................................... 44  
  escapedString ............................................................................................................. 46  
  getCnxState ............................................................................................................... 47  
  getCnxStateString ..................................................................................................... 49  
  getCnxSubStateString ............................................................................................... 50  
  getDefaultDomain ..................................................................................................... 51  
  getErrString .............................................................................................................. 52  
  getHeartbeat ............................................................................................................. 53  
  getHostName .............................................................................................................. 54  
  getPort ....................................................................................................................... 55  
  getReconnectionDelay ............................................................................................. 56  
  getServiceName .......................................................................................................... 57  
  getTimeout ................................................................................................................ 58  
  initializePointCache ................................................................................................. 59  
  isConnected ................................................................................................................. 61  
  isConnecting .............................................................................................................. 62  
  lookupPoint ................................................................................................................ 63
multiplyPointValue ................................................................. 64
openConnection ................................................................. 66
readPoint .............................................................................. 67
registerDomain ..................................................................... 70
registerPoint ......................................................................... 72
retryConnection ..................................................................... 74
sendBinaryPointMessages .................................................... 75
sendLispMessage .................................................................... 76
sendLogin .............................................................................. 79
setConnectionParms ............................................................. 80
setDefaultDomain ............................................................... 82
setHeartbeatTimes ............................................................... 83
setPointLock ......................................................................... 85
setPointSecurity .................................................................... 87
setReconnectionDelay .......................................................... 89
setUsername .......................................................................... 90
shutdown .............................................................................. 91
startHeartbeatTimers ............................................................ 92
startReconnectionTimer ........................................................ 93
unregisterPoint ....................................................................... 94
writeCommand ....................................................................... 96
writePoint .............................................................................. 97

II. Callback Methods ................................................................. 102
onAlive ................................................................................. 103
onAsyncMessage ................................................................. 104
onConnectionFailure ............................................................ 105
onConnectionSuccess ........................................................... 106
onError .................................................................................. 107
onPointChange ...................................................................... 108
onPointEcho .......................................................................... 109
onStatusChange ..................................................................... 110
onSuccess ............................................................................. 111

III. DataHubPoint Methods ......................................................... 113
DataHubPoint .......................................................................... 114
~DataHubPoint ....................................................................... 116
operator= ............................................................................... 117
clear ...................................................................................... 118
copy ....................................................................................... 119
getConfidence ....................................................................... 120
getDateString ......................................................................... 121
getDoubleValue ...................................................................... 122
getFlags ................................................................................ 123
getIntValue ............................................................................ 124
getListeners .......................................................................... 125
getLocked .............................................................................. 126
getName ................................................................................ 127
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getNanoseconds</td>
<td>128</td>
</tr>
<tr>
<td>getQuality</td>
<td>129</td>
</tr>
<tr>
<td>getQualityString</td>
<td>130</td>
</tr>
<tr>
<td>getSeconds</td>
<td>131</td>
</tr>
<tr>
<td>getSecurity</td>
<td>132</td>
</tr>
<tr>
<td>getStringValue</td>
<td>133</td>
</tr>
<tr>
<td>getType</td>
<td>134</td>
</tr>
<tr>
<td>getUserdata</td>
<td>135</td>
</tr>
<tr>
<td>qualifyName</td>
<td>136</td>
</tr>
<tr>
<td>removeListener</td>
<td>138</td>
</tr>
<tr>
<td>setConfidence</td>
<td>139</td>
</tr>
<tr>
<td>setInfo</td>
<td>140</td>
</tr>
<tr>
<td>setLocked</td>
<td>142</td>
</tr>
<tr>
<td>setName</td>
<td>143</td>
</tr>
<tr>
<td>setQuality</td>
<td>144</td>
</tr>
<tr>
<td>setSecurity</td>
<td>145</td>
</tr>
<tr>
<td>setTimeStamp</td>
<td>146</td>
</tr>
<tr>
<td>setFlags</td>
<td>147</td>
</tr>
<tr>
<td>setUserdata</td>
<td>148</td>
</tr>
<tr>
<td>setValue</td>
<td>149</td>
</tr>
<tr>
<td>setValueFromString</td>
<td>151</td>
</tr>
<tr>
<td>unqualifyName</td>
<td>152</td>
</tr>
</tbody>
</table>
Introduction

These three APIs share, as much as possible, common methods and syntax. For this reason they are distributed in one package and documented in a single book.

- **The DataHub API for C++** lets you write programs in C++ that connect to the DataHub over TCP, namely LAN, WAN, or the Internet.
- **The DataHub API for Java** lets you write programs in Java that connect to the DataHub over TCP, namely LAN, WAN, or the Internet.
- **The DataHub API for .NET** lets you write programs in .NET that connect to the DataHub over TCP, namely LAN, WAN, or the Internet. This API is implemented in C#, but can be used with any .NET language.

Preliminaries

The DataHub APIs for C++, Java, and .NET are made up of two classes, `DataHubConnector` and `DataHubPoint` whose methods allow you to interface with the DataHub.

System Requirements

The DataHub APIs for C++, Java, and .NET are compatible with:

- Windows XP Home & Professional
- Windows 2000
- Windows NT 4.0 - All Service Packs should be installed.

Installation

To install the DataHub APIs for C++, Java, and .NET from an archive downloaded from the Cogent web site, follow these steps:

1. Double-click on the program archive `DataHubAPI6.4-xxxxxx-Windows.exe`.
2. Follow the instructions.

To install the DataHub APIs for C++, Java, and .NET from the Cascade Middleware CD, follow these steps:

1. Insert the CD into the drive.
2. Follow the instructions.

DataHub Configuration

The DataHub must be configured to act as a Tunnel/Mirror Master as follows:

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **Tunnel/Mirror**.
3. In the **Tunnelling Master** section, you can configure plain-text or secure tunnelling. Ensure that at least one of these is checked. If you want to change any of the other defaults, please refer to **Tunnel/Mirror in the Cogent DataHub manual** for more information.

   To optimize throughput, un-check the **Try to send data even if it is known to be superseded** option. This will allow the DataHub to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

4. Click **OK** to close the Properties window.

### C++ Programming

The C++ API is intended for application programmers who are working in an unmanaged C++ environment in Windows MFC, Windows ATL, Linux, or QNX.

   To optimize throughput between your program and the DataHub when using the C++ API, you can use the **sendBinaryPointMessages** method.

#### Include Statement for Windows:

```cpp
#include <DataHubConnector.h>
```

#### Include Statement for Linux, QNX4, and QNX6:

```cpp
#include <cogent/DataHubConnector.h>
```

### Java Programming

The Java API implements the **DataHubConnector** class as the basic class used to communicate with the DataHub. Programmers writing stand-alone applications need only the **DataHubConnector** and **DataHubPoint** classes.

#### Java Class Overview

The following classes are included in the Java API installation. They are informally arranged here to give some idea of the interrelationships:
Classes used for general programming

- **DataHubConnector** provides connectivity to the DataHub.
- **DataHubEventConsumer** implements callbacks for DataHubConnector.
- **DataHubEventDispatcher** is an interface that extends DataHubEventConsumer class, and is used by the DataHubBaseApplet class (see below).
- **DataHubPoint** represents DataHub point objects.

Classes used for web programming

- **DataHubBaseApplet** is the applet that makes connections to the DataHub. It provides access to all the data in a single domain. There need be only one DataHubBaseApplet per HTML page, because individual connections are made using DataHubListener (see below). The following two classes extend the DataHubBaseApplet class:
  - **DataHubViewer** displays a table of all the data in the domain. It implements DataHubEventConsumer.
  - **DataHubLink** is used to instantiate a DataHubBaseApplet for supporting DataHubListener widgets. It embeds a small text message "Powered by Cogent" in the page, and its color: red, yellow, or green, indicates the status of the link.
- **DataHubListener** is an applet that gets data from a specific point or points in the DataHub. It is a parasite in the sense that it relies on the connection to the DataHub provided by a DataHubBaseApplet. You can use any number of DataHubListeners per HTML page, without noticeably affecting the rate of data throughput to the page. The following three widgets are extended from DataHubListener:
  - **DataHubLabel** displays the value of a DataHub point.
  - **DataHubEntryField** is an entry field for changing the value of a DataHub point.
  - **DataHubButton** sends a value for a DataHub point.
  - **DataHubToggleButton** toggles a DataHub point between two values.
  - **DataHubCheckBox** toggles a DataHub point between two values.
  - **DataHubRadioButton** A special button, used in a DataHubRadioGroup.
  - **DataHubRadioGroup** A group of DataHubRadioButton that provides a way to select one of several mutually-exclusive values for a DataHub point.
  - **DataHubSlider** changes the value of a DataHub point by sliding a pointer.
  - **DataHubSpinner** sets the value of a DataHub point using up and down arrows.
  - **DataHubProgressBar** gives a graphical representation of the value of a DataHub point.
- **DataHubDummy** is provided as a convenience to the HTML programmer.

For internal use

- **DataHubRendezvous** provides a meeting point for the DataHubBaseApplet and all of the DataHubListener widgets on the page. It is for internal use, providing static da-
ta that gets initialized before any applet starts, giving all the applets a means of finding one another.

**Import Statements**

```java
import cogent.*;
```

or

```java
import cogent.DataHubConnector;
import cogent.DataHubPoint;
```

**.NET Programming**

The .NET API is written in C#, and implements the `DataHubConnector` class. You can compile the file `DHNetAPI.cs` to create a .NET library that can be used by any .NET language, such as Visual Basic .NET.

The .NET installation includes a test program that can be used to connect to an existing DataHub to view data graphically. The two data sets supported are the "DataSim" data generated by the local DataSim program, or the "test" data generated by the Internet data set at http://developers.cogentrts.com. The .NET test program will arrange its graph based on the domain name chosen.

**Requirement Statement**

```java
using Cogent.DataHubAPI;
```

**Example Programs**

There are example programs for the DataHub APIs that ship with the installation archive. If you go to the Windows **Start** menu and choose **Cogent**, you'll see **.Net Test Application** and **C++ Test Application** options. The source code for these applications is here (32-bit/64-bit versions of DataHub):

- `C:\Program Files (x86)\Cogent\API\`
- `C:\Program Files\Cogent\API\`
The DataHubConnector Class

Overview

Syntax

For C++ (Windows MFC, QNX, Linux):

```cpp
class CDataHubConnector : public CWnd
```

For C++ (Windows ATL):

```cpp
class CDataHubConnector : public CWindowImpl<CDataHubConnector,CWindow,CFrameWinTraits>
```

For Java:

```java
public class DataHubConnector
```

For C#:

```csharp
public class DataHubConnector
```

Remarks

This class provides the base functionality for a client to connect to the DataHub. The constructor for this class is `DataHubConnector`. The destructor for this class is `~DataHubConnector`. The methods for this class are arranged by category in the Categorized List of Methods, and alphabetically in the DataHubConnector Methods reference.

Requirement Statements

For C++:

```cpp
#include <CDataHubConnector.h>
```

For Java:

```java
import cogent.DataHubConnector;
```

For C#:

```csharp
using Cogent.DataHubAPI;
```

Categorized List of Methods

These are most of the methods of the DataHubConnector class. The remaining methods, which are used for making callbacks, are presented in the following section.
Status Functions

getCnxState - retrieves the operational state of the connector object.
getCnxStateString - retrieves a string corresponding to the operational state.
getCnxSubStateString - provides detailed information on the connection state (C++ only).
getErrString - retrieves the last error string (C++ only).

Connection Control

setReconnectionDelay - sets the delay time between reconnection attempts.
getReconnectionDelay - retrieves the delay time between reconnection attempts.
startReconnectionTimer - starts the delay timer for reconnection attempts.
cancelReconnectionTimer - stops the delay timer for reconnection attempts.
setHeartbeatTimes - sets the period of the heartbeat and timeout timers.
getHeartbeat - retrieves the heartbeat timer period.
getTimeout - retrieves the timeout timer period.
startHeartbeatTimers - starts the heartbeat and timeout timers.
cancelHeartbeatTimers - cancels the heartbeat and timeout timers.
activeHeartbeatTimers - determines if both heartbeat timers are active.
setConnectionParms - sets the connection parameters.
getHostName - retrieves the host name connection parameter.
getServiceName - retrieves the port service name connection parameter (C++ only).
isConnecting - indicates whether a connection attempt is in progress.
isConnected - indicates whether a connection has been established.
openConnection - attempts to establish a connection to the DataHub.
retryConnection - opens a new connection to the DataHub (C++ only).
closeConnection - closes the connection to the DataHub.
shutdown - prepares for an application shutdown or disconnect (Java and C# only).

Messages

sendLispMessage - sends a message to the DataHub (C++ only).
writeCommand - sends a command to the DataHub (Java and C# only).
escapedString - prepares a string for use with writeCommand (Java and C# only).

Point Handling

initializePointCache - initializes local point cache usage.
lookupPoint - accesses a point from the point cache.
registerDomain - registers to receive updates from domain points.
setDefaultDomain - sets the default domain.
getDefaultDomain - returns the current default domain.
registerPoint - registers to receive updates from a DataHub point.
unregisterPoint - stops receiving updates from a DataHub point.
createPoint - creates a DataHub point.
writePoint - writes a new value to a DataHub point.
readPoint - gets the value of a DataHub point.
setPointLock - sets the lock attributes of a DataHub point.
setPointSecurity - sets the security attributes of a DataHub point.
appendPointValue - appends a string to a DataHub point.
addPointValue - adds a specified amount to a DataHub point value.
multiplyPointValue - multiplies a DataHub point value by a specified amount.
dividePointValue - divides the value of the named point by the specified value.

Making Callbacks

There are several callbacks associated with the DataHubConnector class. In C++ these are methods of the DataHubConnector class itself, while in Java and .NET they are methods of a separate interface, the DataHubEventConsumer interface.

Status Changes

onStatusChange - a virtual method invoked on change of status.

Connections

onConnectionSuccess - a virtual method invoked when a connection is established.
onAlive - a virtual method invoked on receipt of a heartbeat from the DataHub.
onConnectionFailure - a virtual method invoked when a connection or attempt to connect fails.

Message Receipts

onAsyncMessage - a virtual method invoked on receipt of a DataHub message.
onSuccess - a virtual method invoked on receipt of a success message.
onError - a virtual method invoked on receipt of an error message.

Point Changes

onPointChange - a virtual method invoked on receipt of a point value change.
onPointEcho - a virtual method invoked on receipt of a locally changed point value.
The **DataHubPoint Class**

The `DataHubPoint` class represents a DataHub point object.

**Overview**

**Syntax**

**For C++:**

```cpp
class CDataHubPoint
```

**For Java:**

```java
public class DataHubPoint
```

**For C#:**

```csharp
public class DataHubPoint
```

**Remarks**

DataHub point objects are the fundamental objects used to write, receive and manipulate data in the DataHub, via the `DataHubConnector` class. The `DataHubPoint` class provides a rich set of facilities to create, modify and inspect these objects.

DataHub points possess the following properties:

- **value**
  A value whose type is one of `PT_TYPE_STRING`, `PT_TYPE_REAL` (a double) or `PT_TYPE_INT32` (an int). The value is stored in a corresponding format, and can be converted by the various utilities to access the value.

- **quality**
  Indicates whether the DataHub has been updated with actual data or if a point is uninitialized. This is typically either `PT_QUALITY_GOOD` or `PT_QUALITY_BAD`. Connection status can also affect the point quality.

- **confidence**
  A user defined value, typically in the range of 0-100%. This can be used to model 'aging' of a point, and support 'fuzzy math' algorithms.

- **timestamp**
  Tags the real-time origin of the point as it is distributed. Typically this is set by the software module originating the value of a point. It is modelled as seconds and nanoseconds, providing a resolution that is limited only by the OS.

- **userdata**
  Allows the user to associate with a specific point whatever object may be useful to the application. This is primarily used by the point cache capability provided by the...
The DataHubPoint Class

DataHubConnector class (see initializePointCache).

- locked
  Controls access to the point.

- security
  Controls access to the point.

- flags
  For internal use.

Requirement Statements

For C++:

```cpp
#include <CDataHubPoint.h>
```

For Java:

```java
import cogent.DataHubPoint;
```

For C#:

```csharp
using Cogent.DataHubAPI;
```

See also

Categorized List of Methods, DataHubConnector

Categorized List of Methods

Constructors/Destructors

- DataHubPoint - constructs a DataHubPoint object in various ways.
- ~DataHubPoint - destroys a DataHubPoint object.

General Methods

- clear - clears the point.
- getName - retrieves the point name.
- setName - assigns a name to the point.
- qualifyName - creates a point name string qualified by a domain name.
- unqualifyName - removes the domain name qualifier from a point name.

Point Data Access Methods

- getType - retrieves the point data type.
- setValue - sets the point data to the specified type and value.
- setValueFromString - sets the point data to the value represented by a string.
- getDoubleValue - retrieves the point data as a double-typed value.
The DataHubPoint Class

**getIntValue** - retrieves the point data as an int-typed value.
**getStringValue** - retrieves the point data as a string.

**Point Information Access Methods**

**setInfo** - sets the information properties of a point.
**setQuality** - sets the point quality.
**getQuality** - retrieves the point quality.
**getQualityString** - generates a string representing the point quality.
**setConfidence** - sets the user's confidence in a point.
**getConfidence** - retrieves the user's point confidence.
**setTimeStamp** - sets the point timestamp in various ways.
**getSeconds** - retrieves the timestamp seconds component.
**getNanoseconds** - retrieves the timestamp nanoseconds component.
**getDateString** - generates a 'standard' timestamp data/time representation.
**getListeners** - retrieves listeners on the point (Java only).
**removeListener** - removes a listener from the point (Java only).
**setLocked** - sets the locked property of the point.
**getLocked** - retrieves the locked property of the point.
**setSecurity** - sets the security property of the point.
**getSecurity** - retrieves the security property of the point.
**setFlags** - sets the flags property of the point.
**getFlags** - retrieves the flags property of the point.
**setUserdata** - associates the point with a user object.
**getUserdata** - retrieves the user object associated with the point.

**Operators**

**operator=** - assigns a new value to a DataHubPoint object (C++ only).
DHTP - The DataHub Transfer Protocol

DHTP - The DataHub Transfer Protocol

The DataHub Transfer Protocol (DHTP) is used by the DataHub Tunnel/Mirror feature, as well as SkkyHub, ETK, and connected clients to send and receive data in real time over TCP across a LAN, WAN, or the Internet. Originally built upon HTTP, DHTP also supports SSL and WebSocket protocols. In continuous development for over 20 years, DHTP is open and documented in two parts, as the DataHub APIs and the DataHub Command Set.

Each DataHub connected by DHTP requires its own license. License verification is done between DataHubs over the network. Occasionally a slow network may result in misleading "no license" errors. Please refer to TCPLicenseTimeoutSecs for more information.

Additionally, the Cogent DataHub supports various protocols that are native to commonly used industrial applications, like ODBC, OPC, Modbus, etc. The ETK supports OPC UA and Modbus.

Examples

As shown in the above diagram, DHTP may be used for the following connection types:

• DataHub to DataHub for DMZs and tunnelling on LANs and WANs
• DataHub to SkkyHub for OT to IT connections and access to remote locations
• ETK to DataHub for on-premise connections and edge processing
• ETK to SkkyHub for direct connections to the cloud and web HMI
• Custom programs to DataHub, to integrate virtually any application
Applied DHTP Features

SkkyHub and DataHub use DHTP to provide these important Industrial IoT features:

- **Low Bandwidth & Low Latency:** Consumes minimal bandwidth, while functioning with the lowest possible latency
- **Ability to Scale:** Can support hundreds or thousands of interconnected data sources and users
- **Real-Time:** Adds virtually no latency to the data transmission
- **Intelligent Overload Handling:** A broker (DataHub, SkkyHub or ETK) responds appropriately when a data user is unable to keep up with the incoming data rate
- **Quality of Service:** Guarantees consistency of data, preserved through multiple hops

DHTP Protocol Features

DHTP communications between and among Cogent DataHub, SkkyHub, ETK, and their clients meet the following criteria for secure, robust industrial and IIoT data communications:

- **Closed Firewalls:** Keeps all firewall ports closed for both data sources and data users
- **Interoperable Data Format:** Encodes the data so that clients and servers do not need to know each others' protocols
- **Can Daisy Chain Servers:** Multiple instances of brokers (DataHub, SkkyHub or ETK) can be connected to support a wide range of collection and distribution architectures
- **Propagation of Failure Notifications:** Each client application can know with certainty if and when a connection anywhere along the data path has been lost, and when it recovers
- **Simple:** Message syntax is simple enough to be implemented even on resource-constrained devices
- **Streamable:** Messages can be concatenated and streamed without requiring intervening acknowledgements. This allows clients and servers to communicate asynchronously, reducing latency and significantly improving throughput
Appendix A. GNU General Public License

Copyright © 1989, 1991 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:
1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses,
in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that
users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

**Section 3**

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose
of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type “show w”. This is free software, and you are welcome to redistribute it under certain conditions; type “show c” for details.

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something...
other than “show w” and “show c”; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program “Gnomovision” (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.
Appendix B. GNU Lesser General Public License

This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.

Copyright © 1991, 1999 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Version 2.1, February 1999

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code; that you receive source code and can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

We protect your rights with a two-step method:
1. we copyright the library, and
2. we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients
should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the Lesser General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a “work based on the library” and a “work that uses the library”. The former contains code derived from the library, whereas the latter must be combined with the library in order to run.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called “this License”). Each licensee is addressed as “you”.

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

Section 1

You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.
b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.
Section 4

You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

Section 5

A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

Section 6

As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.
You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at runtime a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

Section 7

You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:
a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

Section 8

You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 9

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

Section 10

Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

Section 11

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 12

If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 13

The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

Section 14

If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 15

BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PER-
FORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 16

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

###lt;one line to give the library's name and a brief idea of what it does.###gt; Copyright (C) ###lt;year###gt; ###lt;name of author###gt;

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:
Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

###signature of Ty Coon###, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!
DataHubConnector Methods

These are the methods associated with the DataHubConnector class, listed alphabetically. To see the same methods grouped according to how they are used, please refer to the Categorized List of Methods.
DataHubConnector

DataHubConnector — creates a DataHubConnector object.

Synopsis

For C++:

```cpp
CDataHubConnector();
void;
```

For Java:

```java
DataHubConnector(parent);
DataHubEventDispatcher parent;
```

For C#:

```csharp
DataHubConnector(parent);
DataHubEventConsumer parent;
```

Parameters

`parent`

The parent for this object.

Description

Creates a `DataHubConnector` object.

See Also

`DataHubConnector Class, Categorized List of Methods`
~DataHubConnector

~DataHubConnector — destroys a DataHubConnector object.

Synopsis

For C++:

    ~DataHubConnector();
    void;

For Java, and C#:

    None.

Description

Destroys a DataHubConnector object.

See Also

DataHubConnector Class, Categorized List of Methods
**activeHeartbeatTimers**

*activeHeartbeatTimers* — determines if both heartbeat timers are active.

**Synopsis**

**For C++ and C#:**

```cpp
bool activeHeartbeatTimers();
void;
```

**For Java:**

```java
boolean activeHeartbeatTimers();
void;
```

**Returns**

TRUE if both heartbeat timers are active, FALSE otherwise.

**Description**

This method is used to determine if both heartbeat timers were successfully started. The heartbeat timers are normally started when a connection has been established (see *startHeartbeatTimers*). You may wish to check that the heartbeat mechanism is active before registering a domain or points.

**See Also**

*startHeartbeatTimers*, *cancelHeartbeatTimers*, *setHeartbeatTimes*

**Example**

```cpp
void DataGenerator::onConnectionSuccess (LPCTSTR host, int port)
{
    _super::onConnectionSuccess (host, port); // starts the
    // heartbeat timers

    // proceed to start heartbeat and specify domain
    int domain_flags = DHC_FLAG_REG_FUTURE |
                      DHC_FLAG_REG_QUALIFY |
                      DHC_FLAG_REG_ONCEONLY;

    if (activeHeartbeatTimers()) // started the heartbeat timers
```
if (registerDomain (m_DomainName,
               (DHC_tRegFlags)domain_flags) == ST_OK)
{
    RegisterPoints();
}

addPointValue

addPointValue — adds a specified amount to a DataHub point value.

Synopsis

For C++:

ST_STATUS addPointValue(point, value);

CDataHubPoint& point;
double value;

ST_STATUS addPointValue(pointname, value);

LPCTSTR pointname;
double value;

For Java and C#:

Exception addPointValue(point, value);

DataHubPoint point;
double value;

Exception addPointValue(pointname, value);

String pointname;
double value;

Parameters

point
A DataHubPoint object. The name, seconds and nanoseconds members must be valid.

pointname
The name of the point. The point timestamp is automatically set to the current time.

value
The value to add to the current point value.

Returns

For C++:

- ST_OK if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the onSuccess or onError message handlers.
addPointValue

• **ST_NO_TASK** if a connection to the DataHub does not exist.
• **ST_ERROR** if the connection socket is unable to send the message.

**Description**

Adds the specified value to the current value of the DataHub point. If the DataHub point is not of a numeric type, then the DataHub will respond with an error, and **onError** will be called with the following arguments:

status: **ST_WRONG_TYPE**  
msg: "Wrong type"

If the DataHub point does not exist, then the DataHub will respond with an error, and **onError** will be called with the following arguments:

status: **ST_NO_POINT**  
msg: "Point does not exist"

**Examples**

```
addPointValue(_T("intPoint1"), 1.0);
```

```
CDataHubPoint point;
point.name = "realPoint2";
setPointTimeStamp (&point);
addPointValue(&point, 1.234);
```

**See Also**

appendPointValue, multiplyPointValue, dividePointValue, writePoint
appendPointValue

appendPointValue — appends a string to a DataHub point.

Synopsis

For C++:

```c
ST_STATUS appendPointValue(point, str);
```

```c
CDataHub& point;
LPCTSTR str;

ST_STATUS appendPointValue(pointname, str);

LPCTSTR pointname;
LPCTSTR str;
```

For Java and C#:

```java
Exception appendPointValue(point, str);```

```java
DataHub point;
String str;

Exception appendPointValue(pointname, str);

String pointname;
String str;
```

Parameters

point
A pointer structure of type `DataHubPoint`. The name, seconds and nanoseconds members must be valid.

pointname
The name of the point. The point timestamp is automatically set to the current time.

str
A string to append to the current point string value.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
• **ST_NO_TASK** if a connection to the DataHub does not exist.
• **ST_ERROR** if the connection socket is unable to send the message.

**Description**

This method appends the specified string to the current string value of the DataHub point. If the DataHub point is not a string type (**PT_TYPE_STRING**), then the DataHub will respond with an error, and **onError** will be called with the following arguments:

- **status**: **ST_WRONG_TYPE**
- **msg**: "Wrong type"

If the DataHub point does not exist, then the DataHub will respond with an error, and **onError** will be called with the following arguments:

- **status**: **ST_NO_POINT**
- **msg**: "Point does not exist"

**Examples**

```c
appendPointValue(_T("strPoint1"), _T("this"));
```

```c
CDataHubPoint point;
point.name = "strPoint1";
setPointTimeStamp (&point);
appendPointValue(&point, _T(" and that"));
```

**See Also**

`addPointValue`, `multiplyPointValue`, `dividePointValue`, `writePoint`
cancelHeartbeatTimers

cancelHeartbeatTimers — cancels the heartbeat and timeout timers.

Synopsis

For C++, Java, and C#:

```c
void cancelHeartbeatTimers();
void;
```

Description

See `setHeartbeatTimes` for more details on the timeout timer feature.

See Also

`setHeartbeatTimes, startHeartbeatTimers`
cancelReconnectionTimer

cancelReconnectionTimer — stops the delay timer for reconnection attempts.

Synopsis

For C++, Java, and C#:

```c
void cancelReconnectionTimer();
void;
```

Description

This method cancels the reconnection delay timer. The default behaviour of the `onConnectionSuccess` method is to make a call to `cancelReconnectionTimer`. See `setReconnectionDelay` for more details.
closeConnection

closeConnection — closes the connection to the DataHub.

Synopsis

For C++:

```cpp
void closeConnection();
void;
```

For Java and C#:

```java
void closeConnection(reason);
String reason;
```

Description

This method closes the connection to the DataHub.

See Also

openConnection, retryConnection, isConnecting, isConnected
createPoint

createPoint — creates a DataHub point.

Synopsis

For C++:

```c++
ST_STATUS createPoint(point);
CDataHubPoint& point;
ST_STATUS createPoint(pointname);
LPCTSTR pointname;
```

For Java and C#:

```java
Exception createPoint(point);
DataHubPoint point;
Exception createPoint(pointname);
String pointname;
```

Parameters

- **point**
  - A `DataHubPoint` object. The name member must be valid.

- **pointname**
  - The name of the point.

Returns

For C++:

- **ST_OK** if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- **ST_NO_TASK** if a connection to the DataHub does not exist.
- **ST_ERROR** if the connection socket is unable to send the message.

Description

This method creates a DataHub point. The point quality is set to `PT_QUALITY_BAD` and the timestamp is set to 0. The point type and value are undefined.
See Also

registerPoint, unregisterPoint
dividePointValue

CDataHubConnector: dividePointValue — divides the value of the named point by the specified value.

Synopsis

For C++:

```cpp
ST_STATUS dividePointValue(point, value);

CDataHubPoint& point; double value;

ST_STATUS dividePointValue(pointname, value);

LPCTSTR pointname; double value;
```

For Java and C#:

```java
Exception dividePointValue(point, value);

DataHubPoint point; double value;

Exception dividePointValue(pointname, value);

String pointname; double value;
```

Parameters

- **point**
  A DataHubPoint object. The name, seconds and nanoseconds members must be valid.
- **pointname**
  The name of the point. The point timestamp is automatically set to the current time.
- **value**
  The value by which to divide the current point value.

Returns

For C++:

- ST_OK if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the onSuccess or onError message handlers.
dividePointValue

• ST_NO_TASK if a connection to the DataHub does not exist.
• ST_ERROR if the connection socket is unable to send the message.

Description

This method divides the current value of the DataHub point by the specified value. If the DataHub point is not of a numeric type, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_WRONG_TYPE
msg: "Wrong type"

If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_NO_POINT
msg: "Point does not exist"

Examples

dividePointValue(_T("intPoint1"), 1.0);

CDataHubPoint point;
point.name = "realPoint2";
setPointTimeStamp (&point);
dividePointValue(&point, 1.234);

See Also

appendPointValue, addPointValue, multiplyPointValue, writePoint
escapedString

escapedString — prepares a string for use with writeCommand (Java and C# only).

Synopsis

For Java:

```java
String escapedString(String str, boolean quoted);
String str;
boolean quoted;
String escapedString(String str, boolean quoted, boolean special_only);
String str;
boolean quoted;
boolean special_only;
```

For C#:

```csharp
String escapedString(String str, bool quoted);
String str;
bool quoted;
String escapedString(String str, bool quoted, bool special_only);
String str;
bool quoted;
bool special_only;
```

Parameters

This has not yet been documented.

Returns

This has not yet been documented.

Description

This method prepares a string for use with writeCommand.

See Also

writeCommand
getCnxState

getCnxState — retrieves the operational state of the connector object.

Synopsis

For C++, Java, and C#:

```c
DHC_tState getCnxState();
```

```java
void;
```

```csharp
Returns
```

The current state of the connector object, a member of DHC_tState as defined below.

Description

Retrieves the state of the DataHub connector object. The state is primarily informational, since the user cannot directly change it.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHC_STATE_NONE</td>
<td>This is the initial (0) state, and is immediately changed to DHC_STATE_IDLE within the constructor unless a fundamental initialization error has occurred (eg. Window creation).</td>
</tr>
<tr>
<td>DHC_STATE_IDLE</td>
<td>A connection has never been attempted.</td>
</tr>
<tr>
<td>DHC_STATE_INITIALIZED</td>
<td>Internal initialization completed (by the first attempt to connect) and ready to connect.</td>
</tr>
<tr>
<td>DHC_STATE_CONNECTING</td>
<td>Currently in the process of connecting. See getCnxSubStateString for detailed connection sub-state.</td>
</tr>
<tr>
<td>DHC_STATE_CONNECTING CLOSING</td>
<td>This is a transient internal state used to force closure of an open connection before attempting to reconnect.</td>
</tr>
<tr>
<td>DHC_STATE_CONNECTED</td>
<td>A DataHub connection exists.</td>
</tr>
<tr>
<td>DHC_STATE_ERROR</td>
<td>An error was detected. See getErrString. This is generally a transient state, since the default behaviour is to attempt to reconnect.</td>
</tr>
<tr>
<td>DHC_STATE_RETRY_DELAY</td>
<td>The reconnection timer is active and after which an attempt will be made to reconnect (see setReconnectionDelay).</td>
</tr>
</tbody>
</table>
See Also

`onStatusChange, getCnxStateString, getCnxSubStateString, isConnected, isConnected, getErrString, getReconnectionDelay, setReconnectionDelay`
getCnxStateString

getCnxStateString — retrieves a string corresponding to the operational state.

Synopsis

For C++:

```cpp
CString getCnxStateString();
void;
```

For Java and C#:

```java
String getCnxStateString();
void;
```

Returns

The string corresponding to the operational state of the object (as returned by `getCnxState`).

Description

This method retrieves a string corresponding to the operational state.

See Also

`getCnxState`
getCnxSubStateString

getCnxSubStateString — provides detailed information on the connection state (C++ only).

Synopsis

For C++:

```cpp
CString getCnxSubStateString();
void;
```

Returns

A string containing a short description of the connection status.

Description

This method provides some additional insight into the status of the DataHub connection. A new status string may be obtained on each `onStatusChange`, along with the main `getCnxStateString`, and used for logging or otherwise indicating the connection sequence.

See Also

`onStatusChange` `getCnxStateString`, `getCnxState`
getDefaultDomain

getDefaultDomain — returns the current default domain.

Synopsis

For C++:

    LPCTSTR getDefaultDomain();
    void;

For Java, and C#:

    String getDefaultDomain();
    void;

Returns

    The domain name string, or NULL\(^1\) if none has been set.

Description

    This method accesses the domain name string specified by the setDefaultDomain method. The value returned is based on a local (client) copy, and is therefore not dependent on the connection status.

See Also

    setDefaultDomain

\(^1\)null in Java and C#.
**getErrString**

getErrString — retrieves the last error string (C++ only).

**Synopsis**

For C++:

```cpp
LPCTSTR getErrString ()

void;
```

**Returns**

The string corresponding to the last error detected.

**See Also**

onError
getHeartbeat

getHeartbeat — retrieves the heartbeat timer period.

Synopsis

For C++, Java, and C#:

```cpp
long getHeartbeat();
void;
```

Returns

The period in milliseconds of the heartbeat timer.

Description

See `setHeartbeatTimes` for more details on the heartbeat timer feature.

See Also

`setHeartbeatTimes`, `getTimeout`
getHostName

getHostName — retrieves the host name connection parameter.

Synopsis

For C++:

```cpp
CString getHostName();
void;
```

For Java and C#:

```java
String getHostName();
void;
```

Returns

A string containing the currently set host name.

Description

This method retrieves the host name string set by the `setConnectionParms` command, or an empty string if it has never been set. The value is not affected by `openConnection` or the status of the connection.
getPort

getPort — retrieves the port connection parameter.

Synopsis

For C++, Java, and C#:

```cpp
int getPort();

void;
```

Returns

The connection port number.

Description

This method retrieves the port number set by the `setConnectionParms` command, or 0 if it has never been set. If `setConnectionParms` was used to specify a service name, then the port will be the result of an attempt to convert or look up the servicename specified. The value is not affected by `openConnection` or the status of the connection.
getReconnectionDelay

getReconnectionDelay — retrieves the delay time between reconnection attempts.

Synopsis

For C++, Java, and C#:

```c
long getReconnectionDelay();

void;
```

Returns

The reconnection delay time setting, in milliseconds.

Description

See `setReconnectionDelay` for more details.
getServiceName

getServiceName — retrieves the port service name connection parameter (C++ only).

Synopsis

For C++:

```cpp
CString getServiceName();
void;
```

Returns

A string containing the currently set port service name.

Description

This method retrieves the port service name string set by the `setConnectionParms` command, or an empty string if it has never been set. If `setConnectionParms` was used to set the port directly (as an integer), then the servicename will not have been set. The value is not affected by `openConnection` or the status of the connection.
getTimeout

getTimeout — retrieves the timeout timer period.

Synopsis

For C++, Java, and C#:

```c
long getTimeout();
void;
```

Returns

The period in milliseconds of the timeout timer.

Description

See `setHeartbeatTimes` for more details on the timeout timer feature.

See Also

`setHeartbeatTimes`, `getHeartbeat`
**initializePointCache**

*initializePointCache* — initializes local point cache usage.

**Synopsis**

For C++:

```c
ST_STATUS initializePointCache();

void;
```

For Java, and C#:

```java
Exception initializePointCache();

void;
```

**Returns**

For C++:

`ST_OK` if the point cache was successfully initialized, otherwise `ST_ERROR`.

**Description**

The point cache is a list of all points received. It is automatically built and maintained as each point update is received from the DataHub. In order for a point to exist in the cache, a `readPoint`, `registerPoint`, or `registerDomain` was required to generate the point update. The value, type, and timestamp of the cached points is updated, but the point itself is persistent.

The point cache allows the user to associate data with the point (through its `m_userdata` member) and quickly access it when the point is updated (from `onPointChange`). The point cache also provides a way to synchronously read a current point value, avoiding the `readPoint` method which provides the point value asynchronously.

**Example 1**

This example illustrates use of the point cache to efficiently update a user interface control with the latest point value.

```c
// during initialization, we register for all points
registerDomain("myDomain",
    DHC_FLAG_REG_FUTURE | DHC_FLAG_ONCEONLY_FUTURE);
```

```java
// during initialization, we register for all points
registerDomain("myDomain",
    DHC_FLAG_REG_FUTURE | DHC_FLAG_ONCEONLY_FUTURE);
```
// associate an MFC control with a point
// perhaps this is called on user entering a desired point name
// (don't forget to clear the previously associated point userdata!)
void onUiPointNameChanged(LPCTSTR pointname, CStatic *pValueWnd)
{
    DataHubPoint *ppoint = lookupPoint(pointname);
    if (ppoint)
    {
        ppoint->m_userdata = pointname;
    }
}

void onPointChange(DataHubPoint point)
{
    if (point.m_userdata)
    {
        // update the MFC control
        CStatic *pTxt = point.m_userdata;
        CString str;
        str.Format("%f", point.getDoubleValue());
        pTxt->SetWindowText(str);
    }
}

Example 2

This example illustrates use of the point cache to provide immediate access to a point value.

....

// part of the initialization code
registerDomain("myDomain",
    DHC_FLAG_REG_FUTURE | DHC_FLAG_ONCEONLY_FUTURE);
....

// read point value:
DataHubPoint *point = lookupPoint("IntPoint1");
int ival = point->getIntValue();
....

See Also

registerPoint, unregisterPoint, createPoint
**isConnected**

isConnected — indicates whether a connection has been established.

**Synopsis**

For C++ and C#:

```cpp
bool isConnected();
void;
```

For Java:

```java
boolean isConnected();
void;
```

**Returns**

TRUE is a connection to the DataHub has been established, FALSE otherwise.

**See Also**

isConnecting, openConnection, retryConnection, closeConnection
isConnecting

isConnecting — indicates whether a connection attempt is in progress.

Synopsis

For C++ and C#:

```cpp
bool isConnecting();

void;
```

For Java:

```java
boolean isConnecting();

void;
```

Returns

TRUE if the object is currently in the process of establishing a connection with the DataHub, FALSE otherwise.

See Also

isConnected, openConnection, retryConnection, closeConnection
lookupPoint

lookupPoint — accesses a point from the point cache.

Synopsis

For C++:

    DataHubPoint* lookupPoint(pointname);
    LPCTSTR pointname;

For Java, and C#:

    DataHubPoint lookupPoint(pointname);
    String pointname;

Parameters

    pointname
        The name of the point.

Returns

    The corresponding point object in the point cache.

Description

    This method returns a pointer to a cached DataHubPoint object matching the specified name object. If no object matching the name is in the cache, then NULL is returned.

See Also

    initializePointCache
multiplyPointValue

— multiplies a DataHub point value by a specified amount.

Synopsis

For C++:

```cpp
ST_STATUS multiplyPointValue(point, value);
CDataHubPoint& point;
double value;
ST_STATUS multiplyPointValue(pointname, value);
LPCTSTR pointname;
double value;
```

For Java and C#:

```java
Exception multiplyPointValue(point, value);
DataHubPoint point;
double value;
Exception multiplyPointValue(pointname, value);
String pointname;
double value;
```

Parameters

- **point**
  A `DataHubPoint` object. The name, seconds and nanoseconds members must be valid.

- **pointname**
  The name of the point. The point timestamp is automatically set to the current time.

- **value**
  The value by which to multiply the current point value.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
• **ST_NO_TASK** if a connection to the DataHub does not exist.
• **ST_ERROR** if the connection socket is unable to send the message.

**Description**

This method multiplies the current value of the DataHub point by the specified value. If the DataHub point is not of a numeric type, then the DataHub will respond with an error, and `onError` will be called with the following arguments:

```plaintext
status: ST_WRONG_TYPE
msg: "Wrong type"
```

If the DataHub point does not exist, then the DataHub will respond with an error, and `onError` will be called with the following arguments:

```plaintext
status: ST_NO_POINT
msg: "Point does not exist"
```

**Examples**

```plaintext
multiplyPointValue(_T("intPoint1"), 1.0);
```

```plaintext
CDataHubPoint point;
point.name = "realPoint2";
setPointTimeTimestamp (&point);
multiplyPointValue(&point, 1.234);
```

**See Also**

`appendPointValue`, `addPointValue`, `dividePointValue`, `writePoint`
openConnection

openConnection — attempts to establish a connection to the DataHub.

Synopsis

For C++, Java, and C#:

```c
void openConnection();
void;
```

Description

This method attempts to establish a connection to the DataHub, using the parameters set by `setConnectionParms`.

See Also

`retryConnection, closeConnection, isConnected, isConnected`
readPoint

readPoint — gets the value of a DataHub point.

Synopsis

For C++:

```cpp
ST_STATUS readPoint(point, create, force);

CDataHubPoint& point;
bool create=TRUE;
bool force=TRUE;

ST_STATUS readPoint(pointname, create, force);

LPCTSTR pointname;
bool create=TRUE;
bool force=TRUE;
```

For Java:

```java
Exception readPoint(point);

DataHubPoint point;

Exception readPoint(point, create, force);

DataHubPoint point;
boolean create;
boolean force;

Exception readPoint(pointname);

String pointname;

Exception readPoint(pointname, create, force);

String pointname;
boolean create;
boolean force;
```

For C#:

```csharp
Exception readPoint(point);

DataHubPoint point;

Exception readPoint(point, create, force);
```
DataHubPoint point;
bool create;
bool force;

Exception readPoint(pointname);

String pointname;

Exception readPoint(pointname, create, force);

String pointname;
bool create;
bool force;

Parameters

point
  A DataHubPoint object. The name member must be valid.

pointname
  The name of the point.

create
  If TRUE, then the point is created if it does not already exist. If FALSE and the point does not exist, then an error message will be generated.

force
  If a valid connection to the DataHub exists but the message is undeliverable immediately (due to the TCP buffer being full), then if the force parameter is TRUE, the message is queued and will be transmitted when possible.

Returns

For C++:

- ST_OK if the command was successfully sent to the DataHub.
- ST_NO_TASK if a connection to the DataHub does not exist.
- ST_ERROR if the connection socket is unable to send the message.

Description

This method requests that the value of the specified point be transmitted. The point value is received as a point message, which causes the virtual method onPointChange to be called. The point value cannot be obtained synchronously.

If the create parameter is FALSE and the point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_NO_POINT
msg: "Point does not exist"
Examples

```cpp
readPointValue(_T("strPoint1"));

CDataHubPoint point;
point.name = "strPoint1";
readPoint(&point, FALSE);
```

See Also

`writePoint`
registerDomain

registerDomain — registers to receive updates from domain points.

Synopsis

For C++:

ST_STATUS registerDomain(domainname, flags);

LPCTSTR domainname;
int flags;

For Java and C#:

Exception registerDomain(domainname, flags);

String domainname;
int flags;

Parameters

domainname

The name of a domain.

flags

Specifies conditions and actions associated with registering the domain points, as fol-

lows (flags may be combined):

• DHC_FLAG_REG_FUTURE: points created in the future are also affected. If not set,
then only those points existing (in the domain) at the time the command message is
received by the DataHub are registered.

• DHC_FLAG_REG_QUALIFY: point names are to be transmitted with the domain
name prepended to the point name, as domainname:pointname.

• DHC_FLAG_REG_ONCEONLY: the point values are transmitted only once, and the
points remain unregistered. This mode is useful for obtaining an initial list of avail-
able points from which a selection is made of which ones to register.

Returns

For C++:

• ST_OK if the command was successfully sent to the DataHub. Since the command is
sent asynchronously, the actual success or failure of the command must be determined
through the onSuccess or onError message handlers.

• ST_NO_TASK if a connection to the DataHub does not exist.

• ST_ERROR if the connection socket is unable to send the message.
registerDomain

Description

This method registers the client to receive updates from the DataHub when the value of any points in the specified domain change. The `onPointChange` method is called for each point update received. The `registerDomain` method may be called more than once, and on different domains.

Upon registering the domain, all points currently in that domain of the DataHub are immediately transmitted to the client. Points that are subsequently created will only be sent if the `DHC_FLAG_REG_FUTURE` flag is specified. The `DHC_FLAG_REG_QUALIFY` flag is useful for distinguishing points from different domains that have the same name.

In a typical scenario, either there is no need to register for all points, or the point names are not even known. In either case, you can use the `DHC_FLAG_REG_ONCEONLY` flag to generate an initial list of all available points in the specified domain. Then you can call `registerPoint` on any points of interest, as they are received in `onPointChange`.

See Also

`unregisterPoint`, `setDefaultDomain`, `registerPoint`
registerPoint

registerPoint — registers to receive updates from a DataHub point.

Synopsis

For C++:

```cpp
ST_STATUS registerPoint(point, create);
CDataHubPoint& point;
bool create=TRUE;
ST_STATUS registerPoint(pointname, create);
LPCTSTR pointname;
bool create=TRUE;
```

For Java:

```java
Exception registerPoint(point, create);
DataHubPoint point;
boolean create;
Exception registerPoint(pointname, create);
String pointname;
boolean create;
```

For C#:

```csharp
Exception registerPoint(point, create);
DataHubPoint point;
bool create;
Exception registerPoint(pointname, create);
String pointname;
bool create;
```

Parameters

- **point**
  
  A `DataHubPoint` object. The name member must be valid.

- **pointname**
  
  The name of the point.
create

If TRUE, then the point is created if it does not already exist. If FALSE and the point does not exist, then an error message will be generated.

Returns

For C++:

- **ST_OK** if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- **ST_NO_TASK** if a connection to the DataHub does not exist.
- **ST_ERROR** if the connection socket is unable to send the message.

Description

When a point is registered with the DataHub, any changes to the value of that point in the DataHub will cause the `onPointChange` method to be called. If the DataHub point does not exist, then the DataHub will respond with an error, and `onError` will be called with the following arguments:

- **status**: ST_NO_POINT
- **msg**: "Point does not exist"

See Also

- `writePoint`
- `unregisterPoint`
- `createPoint`
- `registerDomain`
retryConnection

retryConnection — opens a new connection to the DataHub (C++ only).

Synopsis

For C++:

    void retryConnection();
    void;

Description

This method attempts to reestablish a connection to the DataHub.

See Also

openConnection, closeConnection, isConnected, isConnected
sendBinaryPointMessages

sendBinaryPointMessages — formats data in binary mode (C++ only).

Synopsis

For C++:

```
ST_STATUS sendBinaryPointMessages(enable);
```

```
bool enable;
```

Parameters

```
enable
```

```
TRUE enables binary mode, FALSE disables it.
```

Returns

- **ST_OK** if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- **ST_NO_TASK** if no connection to the DataHub is established.
- **ST_TOO_LARGE** if the message string exceeds the message buffer size.
- **ST_ERROR** if the format contains an error or if the connection socket is unable to send the message.

Description

This method tells the CDataHubConnector instance to format data in binary mode, and also to request binary mode transmissions from the DataHub. Binary messages are more CPU efficient than ASCII messages.
**sendLispMessage**

*sendLispMessage* — sends a message to the DataHub (C++ only).

**Synopsis**

For C++:

```c
ST_STATUS sendLispMessage(force, format, ...);
```

```c
bool force;
char* format;
...;
```

**Parameters**

*force*

If a valid connection to the DataHub exists but the message is undeliverable immediately (due to the TCP buffer being full), then if the force parameter is TRUE, the message is queued and will be transmitted when possible.

*format*

The text formatting string (see below).

... Parameters required by the specified format.

**Returns**

For C++:

- **ST_OK** if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- **ST_NO_TASK** if no connection to the DataHub is established.
- **ST_TOO_LARGE** if the message string exceeds the message buffer size.
- **ST_ERROR** if the format contains an error or if the connection socket is unable to send the message.

**Description**

This method is used to format and send commands to the DataHub, and should only be used by those very familiar with the operation of the DataHub. The DataHub command set is described in the Using Commands chapter of the Cogent DataHub manual. To send a command to the DataHub, it must be formatted such that strings and control characters are preserved through the message transfer and delivery process. The specialized format control specifiers of sendLispMessage make this easy to do.
The writeCommand method gives a similar functionality for Java and C#.

The format control string is similar to that used by printf, with the following differences:

- The %A field type specifier escapes all occurrences of double quotes, i.e., substitutes " for each occurrence of ".
- The %a field type specifier escapes all occurrences of the following characters in addition to the double quote: \ CR LF FF TAB ( ).
- The %s field type specifier escapes the same characters as %a, and also encloses the string in double quotes.
- The %d (or %i) field type specifier assumes a parameter of type int, or of type long if preceded by an 'l' type specifier (%ld). No other type length specifiers are supported.
- The %f and %g field type specifiers assume a parameter of type double. No type length specifiers are supported.
- Other field type specifiers such as %c, %p, %n, %o, %u, %x and %e are not supported.
- The '*' field width specifier is not supported.

The string parameters corresponding to the %s, %a or %A field type specifiers must be of type char *.

Example

```c
void CSetpoint::BuildObjectHierarchy(LPCTSTR sDomain, LPCTSTR sAssembly)
{
    CT2A aDomain(sDomain);
    char *domain = aDomain;
    CT2A aAssembly(sAssembly);
    char *assembly = aAssembly;
    CT2A aName(sName);
    char *name = aName;

    // (assembly domain name)
    pCnx->sendLispMessage(TRUE, "(assembly %s CSetpoint)", domain);
    // (subassembly domain assemblyname subassemblyname instancename)
    pCnx->sendLispMessage(TRUE, "(subassembly %s %s CSetpoint %s)",
                           domain, assembly, name);
    // (property domain attrname propid propname type rw dflt_value dflt_conf)
    pCnx->sendLispMessage(TRUE, "(property %s CSetpoint auto
                           AutoMode int rw 0 100)", domain);
    pCnx->sendLispMessage(TRUE, "(property %s CSetpoint auto
                           AutoMode int rw 0 100)", domain);
```
sendLispMessage

```lisp
AutoTime r8 rw 0 100", domain);
}
```
sendLogin

sendLogin — transmits the user name and password.

Synopsis

For C++:

```cpp
void sendLogin();
void;
```

For Java and C#:

```java
public void sendLogin();
void;
```

Description

This method transmits the user name and password previously set by a call to `setName` or `setConnectionParms`. The `sendLogin` method is normally called automatically when a successful connection is made to the DataHub, prior to the `onConnectionSuccess` user callback.
setConnectionParms

setConnectionParms — sets the connection parameters.

Synopsis

For C++:

```cpp
void setConnectionParms(hostname, servicename);

LPCTSTR hostname;
LPCTSTR servicename;

void setConnectionParms(hostname, port);

LPCTSTR hostname;
int port;

void setConnectionParms(hostname, port, username, password);

LPCTSTR hostname;
int port;
LPCTSTR username;
LPCTSTR password;
```

For Java and C#:

```java
void setConnectionParms(hostname, port);

String hostname;
int port;

void setConnectionParms(hostname, port, username, password);

String hostname;
int port;
String username;
String password;
```

Parameters

- **hostname**
  - The name of the host running the DataHub.
- **servicename**
  - The name of the port on which to connect.
- **port**
  - The connection port number.
**username**

The name of a user.

**password**

A password for that user.

**Description**

This method sets the connection parameters to be used by the `openConnection` method. The port may be specified as either a string or directly by its number. The string may represent the port number, which is simply converted to an integer, or the symbolic port servicename. If a servicename is specified, then a lookup is performed immediately, and the resulting port number may be verified by following with a call to `getPort`.

The expanded syntax allows for setting the user name and password. See `setUsername` for more details.
setDefaultDomain

setDefaultDomain — sets the default domain.

Synopsis

For C++:

```c
ST_STATUS setDefaultDomain(domainname);
```
```
LPCTSTR domainname;
```

For Java and C#:

```java
Exception setDefaultDomain(domainname);
```
```
String domainname;
```

Parameters

domainname

The name of the domain.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is
  sent asynchronously, the actual success or failure of the command must be determined
  through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method sets the name of the DataHub domain to be automatically associated with
all unqualified point names (i.e., point names that do not specify a domain). Since in many
applications there is a single domain to which most if not all points belong, this feature
makes point references simpler and shorter. Points in other domains can still be refer-
enced by qualifying the point name with a specific domain name (see `qualifyName`).

If the DataHub is not connected at the time this method is called, then the domain will be
transmitted when the connection is established.

See Also

`getDefaultDomain`, `qualifyName`
setHeartbeatTimes

setHeartbeatTimes — sets the period of the heartbeat timers.

Synopsis

For C++:

```c++
ST_STATUS setHeartbeatTimes(heartbeat_ms, timeout_ms);
long heartbeat_ms;
long timeout_ms;
```

For Java and C#:

```java
Exception setHeartbeatTimes(heartbeat_ms, timeout_ms);
long heartbeat_ms;
long timeout_ms;
```

Parameters

- `heartbeat_ms`
  - The period in milliseconds for the heartbeat timer, or 0 if a timer will not be used.
- `timeout_ms`
  - The period in milliseconds for the timeout timer, or 0 if a timer will not be used.

Returns

For C++:

- `ST_OK` if the timer values accepted, otherwise `ST_ERROR`. `ST_ERROR` can occur for the following reasons:
  - The timeout timer value is not greater than heartbeat (if both values are greater than 0).
  - The timers were already active and either timer failed to restart.

Description

The DataHub and API provide two heartbeat services to ensure the integrity of the connection:

1. A local, internal timeout timer, which when it fires, will close the connection and generate an error.
2. A periodic 'alive' heartbeat message from the DataHub, which triggers the virtual method `onAlive`.
The timers can be started with `startHeartbeatTimers` once the respective periods have been set. If a timer is already running when `setHeartbeatTimes` is called, then that timer is restarted with the new period. If the timer is already running and the specified period is 0, then the timer is stopped.

The timeout timer is automatically reset whenever there is any activity over the connection, including the heartbeat timer message.

**See Also**

`getHeartbeat`, `getTimeout`, `startHeartbeatTimers`, `cancelHeartbeatTimers`
setPointLock

setPointLock — sets the lock attributes of a DataHub point.

Synopsis

For C++:

```cpp
ST_STATUS setPointLock(point, locked);
CDataHubPoint& point;
bool locked;
ST_STATUS setPointLock(pointname, locked);
LPCTSTR pointname;
bool locked;
```

For Java:

```java
Exception setPointLock(point, locked);
DataHubPoint point;
boolean locked;
Exception setPointLock(pointname, locked);
String pointname;
boolean locked;
```

For C#:

```csharp
Exception setPointLock(point, locked);
DataHubPoint point;
bool locked;
Exception setPointLock(pointname, locked);
String pointname;
bool locked;
```

Parameters

- **point**
  
  A `DataHubPoint` object. The name and security members must be valid.

- **pointname**
  
  The name of the point.
setPointLock

locked

If `TRUE`, the point will have its locked attribute set, otherwise the locked attribute is cleared.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method sets the lock attributes of a DataHub point.

See Also

`setPointSecurity`
setPointSecurity

setPointSecurity — sets the security attributes of a DataHub point.

Synopsis

For C++:

```cpp
ST_STATUS setPointSecurity(point, security);

CDataHubPoint& point;
int security;
ST_STATUS setPointSecurity(pointname, security);

LPCTSTR pointname;
int security;
```

For Java and C#:

```java
Exception setPointSecurity(point, security);

CDataHubPoint& point;
int security;
Exception setPointSecurity(pointname, security);

String pointname;
int security;
```

Parameters

- **point**
  A DataHubPoint object. The name and security members must be valid.
- **pointname**
  The name of the point.
- **security**
  The new security level to which the DataHub point will be set.

Returns

For C++:

- **ST_OK** if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the onSuccess or onError message handlers.
- **ST_NO_TASK** if a connection to the DataHub does not exist.
• **ST_ERROR** if the connection socket is unable to send the message.

**Description**

This method sets the security attributes of a DataHub point. An update to a DataHub point will be rejected unless the security level associated with the new point value is greater than or equal to the corresponding DataHub point. The default DataHub point security level is 0.

**See Also**

*setPointLock*
**setReconnectionDelay**

setReconnectionDelay — sets the delay time between reconnection attempts.

**Synopsis**

For C++, Java, and C#:

```c
void setReconnectionDelay(recon_ms);
```

```c
long recon_ms;
```

**Parameters**

- `recon_ms`
  
  The delay time, in milliseconds, after connection failure before another attempt is automatically made.

**Description**

The `DataHubConnector` class provides an internal facility to automatically attempt reconnection in event of a failure. The default `onConnectionFailure` method will trigger the reconnection timer.

When the timer fires, the `retryConnection` method is called to cancel the timer and attempt to connect.

**See Also**

`getReconnectionDelay`, `startReconnectionTimer`, `cancelReconnectionTimer`
**setUsername**

setUsername — stores a user name and password for this connection.

**Synopsis**

For C++:

```c
void setUsername(username, password);

LPCTSTR username;
LPCTSTR password;
```

For Java and C#:

```java
public void setUsername(username, password);

String username;
String password;
```

**Parameters**

- **username**
  - The name of a user.
- **password**
  - A password for that user.

**Description**

This method stores a user name and password for this connection, to be transmitted by a subsequent call to `sendLogin`. When a subsequent successful connection is made to the DataHub, `sendLogin` will be called prior to the `onConnectionSuccess` callback. The .Net and Java implementations of `setUsername` will disconnect from the DataHub if the connection is currently active. The C++ implementation will not. If either of the username or password parameters is "" or NULL, then no username or password will be sent on the next successful connection.

**See Also**

`sendLogin, setConnectionParms`
**shutdown**  

*shutdown* — prepares for an application shutdown or disconnect (Java and C# only).

**Synopsis**

For Java, and C#:

```java
void shutdown(reason);

String reason;
```

**Parameters**

*reason*  
A character string that may be passed to the *onConnectionFailure* callback indicating the reason for the shutdown. If *reason* is null, a default string will be used.

**Description**

This method cleans up all resources, timers and sockets associated with the connection in preparation for application shutdown or removal of the *DataHubConnector* object. The application should not call any methods of the object or access any of its members after this method call is made. The result of any access to the object after this call is undefined.

**See Also**

*onConnectionFailure*
gesture


testHeartbeatTimers

testHeartbeatTimers — starts the heartbeat and timeout timers.

Synopsis

For C++:

ST_STATUS testHeartbeatTimers();

void;

For Java and C#:

Exception testHeartbeatTimers();

void;

Returns

For C++:

ST_OK if the timers were successfully started, otherwise ST_ERROR.

Description

This method is used to start the two heartbeat timers. If a timer period has been set to 0, then that heartbeat timer function is disabled. The default behaviour of the onConnectionSuccess method is to make a call to testHeartbeatTimers.

See Also

setHeartbeatTimes, cancelHeartbeatTimers, onConnectionSuccess
**startReconnectionTimer**

*startReconnectionTimer* — starts the delay timer for reconnection attempts.

**Synopsis**

**For C++, Java, and C#:**

```c
void startReconnectionTimer();
void;
```

**Description**

This method is used to start the reconnection timer. The default behaviour of the *onConnectionFailure* method is to make a call to *startReconnectionTimer*. See *setReconnectionDelay* for more details.
unregisterPoint

unregisterPoint — stops receiving updates from a DataHub point.

Synopsis

For C++:

```c
ST_STATUS unregisterPoint(point);
CDataHubPoint & point;
ST_STATUS unregisterPoint(pointname);
LPCTSTR pointname;
```

For Java and C#:

```c
Exception unregisterPoint(point);
DataHubPoint point;
Exception unregisterPoint(pointname);
String pointname;
```

Parameters

- point
  
  A DataHubPoint object. The name member must be valid.

- pointname

  The name of the point.

Returns

For C++:

- ST_OK if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the onSuccess or onError message handlers.
- ST_NO_TASK if a connection to the DataHub does not exist.
- ST_ERROR if the connection socket is unable to send the message.

Description

This method stops changes to the specified DataHub point from invoking the onPointChange method. If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:
status: ST_NO_POINT
msg: "Point does not exist"

See Also

writePoint, registerPoint
writeCommand

writeCommand — sends a command to the DataHub (Java and C# only).

Synopsis

For Java and C#:

```java
Exception writeCommand(String command);
String command;
```

Parameters

`command`

A concatenation of `escapedStrings` whose original format was a Lisp command.

Returns

This has not yet been documented.

Description

This method is used to send formatted commands to the DataHub, and should only be used by those very familiar with the operation of the DataHub. The DataHub command set is described in the Using Commands chapter of the Cogent DataHub manual. To send a command to the DataHub, it must be formatted such that strings and control characters are preserved through the message transfer and delivery process. You must concatenate together the necessary command string, using the `escapedString` method to provide the necessary protection of strings.

The `sendLispMessage` method gives a similar functionality for C++.

See Also

`escapedString`
writePoint

writePoint — writes a new value to a DataHub point.

Synopsis

For C++:

```c
ST_STATUS writePoint(point, create, force);
```

```c
CDataHubPoint& point;
bool create=TRUE;
int force=FALSE;
```

```c
ST_STATUS writePoint(pointname, value, create, force);
```

```c
LPCTSTR pointname;
int value;
bool create=TRUE;
bool force=FALSE;
```

```c
ST_STATUS writePoint(pointname, value, create, force);
```

```c
LPCTSTR pointname;
double value;
bool create=TRUE;
bool force=FALSE;
```

```c
ST_STATUS writePoint(pointname, value, create, force);
```

```c
LPCTSTR pointname;
LPCTSTR value;
bool create=TRUE;
bool force=FALSE;
```

For Java:

```java
Exception writePoint(point, create, force);
```

```java
DataHubPoint point;
boolean create;
boolean force;
```

```java
Exception writePoint(pointname, value, create, force);
```

```java
String pointname;
String value;
boolean create;
```
boolean force;
Exception writePoint(pointname, value);

String pointname;
String value;

Exception writePoint(pointname, value, create, force);

String pointname;
double value;
boolean create;
boolean force;

Exception writePoint(pointname, value);

String pointname;
double value;

Exception writePoint(pointname, value, create, force);

String pointname;
int value;
boolean create;
boolean force;

Exception writePoint(pointname, value);

String pointname;
int value;

Exception writePoint(pointname, type, value_as_string, create, force);

String pointname;
int type;
String value_as_string;
boolean create;
boolean force;

For C#:

Exception writePoint(point, create, force);

DataHubPoint point;
bool create;
bool force;

Exception writePoint(pointname, value, create, force);

String pointname;
String value;
bool create;
bool force;

Exception writePoint(pointname, value);

String pointname;
String value;

Exception writePoint(pointname, value, create, force);

String pointname;
double value;
bool create;
bool force;

Exception writePoint(pointname, value);

String pointname;
double value;

Exception writePoint(pointname, value, create, force);

String pointname;
int value;
bool create;
bool force;

Exception writePoint(pointname, value);

String pointname;
int value;

Exception writePoint(pointname, type, value_as_string, create, force);

String pointname;
int type;
String value_as_string;
bool create;
bool force;

**Parameters**

**point**

A DataHubPoint object. The name, type, value, seconds and nanoseconds members must be valid.

**create**

If TRUE, then the point is created if it does not already exist. If FALSE and the point does not exist, then an error message will be generated.
force
If a valid connection to the DataHub exists but the message is undeliverable immediately (due to the TCP buffer being full), then if the force parameter is TRUE, the message is queued and will be transmitted when possible.

pointname
The name of the point. The point timestamp is automatically set to the current time.

value
The integer, double, or string value to write to the point. The type of the point in the DataHub will be changed accordingly.

Returns
For C++:
• ST_OK if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the onSuccess or onError message handlers.
• ST_NO_TASK if a connection to the DataHub does not exist.
• ST_ERROR if the connection socket is unable to send the message.

Description
This method writes the given point to the DataHub. If a DataHubPoint is used, then the point name, type, value, quality and timestamp members must be set. If the point is specified by name, then the type is determined by the specific overloaded method and the timestamp is set to the current time.

If a domain name has been associated with the client (see setDefaultDomain), then the point will be written to that domain, otherwise it will be written to the domain named default. In any case, the domain can always be overridden by qualifying the point name as domain name:point name (see qualifyName).

If the point has been registered (see registerPoint), and the DataHub point is successfully written, then the onPointEcho method will be invoked.

If the create parameter is FALSE and the point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_NO_POINT
msg: "Point does not exist"

Examples

```
CDataHubPoint point;
point.name = "point1";
point.type = PT_TYPE_REAL;
point.quality = PT_QUALITY_GOOD;
```
point.value = 123.456;
setPointTimeStampl(&point);
// point must exist or error is generated
writePoint(&point, FALSE);
// write real-valued point, create if needed
writePoint(_T("point2"), 123.456);

See Also

readPoint
Callback Methods

These are the callback methods associated with the `DataHubConnector` class. For C++, they are methods of the `DataHubConnector` class itself. For Java and C#, they are methods of the `DataHubEventConsumer` interface.
onAlive

onAlive — a virtual method invoked on receipt of a heartbeat from the DataHub.

Synopsis

For C++, Java, and C#:

    void onAlive();

    void;

Description

This method is invoked upon receipt of the 'alive' heartbeat message from the DataHub. See `setHeartbeatTimes` for more details on this feature.

See Also

`setHeartbeatTimes`, `getHeartbeat`, `startHeartbeatTimers`, `cancelHeartbeatTimers`
onAsyncMessage

onAsyncMessage — a virtual method invoked on receipt of a DataHub message.

Synopsis

For C++:

```cpp
virtual void onAsyncMessage(argc, argv);
int argc;
char** argv;
```

For Java and C#:

```java
virtual void onAsyncMessage(arguments);
String[] arguments;
```

Parameters

`argc`

The number of parameters in the DataHub message.

`argv, arguments`

An array of strings which are the parameters in the DataHub message.

Description

This virtual method is invoked on receipt of a DataHub message. This method is only called if none of the other callbacks are called.

See Also

`onStatusChange, onSuccess, onError, onConnectionSuccess, onConnectionFailure`
onConnectionFailure

onConnectionFailure — a virtual method invoked when a connection or attempt to connect fails.

Synopsis

For C++:

    virtual void onConnectionFailure(host, port, reason);

    LPCTSTR host;
    int port;
    LPCTSTR reason;

For Java and C#:

    virtual void onConnectionFailure(host, port, reason);

    String host;
    int port;
    String reason;

Parameters

    host
    The name of the host running the DataHub.

    port
    The connection port number.

    reason
    The reason for the failure.

Description

This virtual method is invoked when a connection or attempt to connect fails. The default behaviour is to make a call to startReconnectionTimer.

See Also

    onStatusChange, onConnectionSuccess, startReconnectionTimer
onConnectionSuccess

onConnectionSuccess — a virtual method invoked when a connection is established.

Synopsis

For C++:

```cpp
virtual void onConnectionSuccess(host, port);

LPCTSTR host;
int port;
```

For Java and C#:

```java
virtual void onConnectionSuccess(host, port);

String host;
int port;
```

Parameters

- **host**
  - The name of the host running the DataHub.
- **port**
  - The connection port number.

Description

This virtual method is invoked when a connection is established. The default behaviour is to make a call to `cancelReconnectionTimer`.

See Also

- `onStatusChange`, `onConnectionFailure`, `cancelReconnectionTimer`
onError

onError — a virtual method invoked on receipt of an error message.

Synopsis

For C++:

```c++
virtual void onError (status, err_str, cmd, parms);
```

```c++
ST_STATUS status;
LPCTSTR err_str;
LPCTSTR cmd;
LPCTSTR parms;
```

For Java and C#:

```java
virtual void onError (status, err_str, cmd, parms);
```

```java
int status;
String err_str;
String cmd;
String parms;
```

Parameters

- **status**
  The status code returned by the DataHub.

- **err_str**
  The error string, providing more detailed information about the error.

- **cmd**
  The original command sent to the DataHub to which this reply corresponds.

- **parms**
  The list of parameters sent as part of the command, as a single space-separated string, or NULL\(^1\) if none are returned. If the command involved multiple parameters, the list may be truncated.

Description

This virtual method is invoked on receipt of an error message.

See Also

- `getErrString`, `onSuccess`, `onConnectionSuccess`, `onConnectionFailure`\(^1\)

---

\(^1\) null in Java and C#.
onPointChange

onPointChange — a virtual method invoked on receipt of a point value change.

Synopsis

For C++:

```cpp
virtual void onPointChange(point);
CDataHubPoint& point;
```

For Java and C#:

```java
virtual void onPointChange(point);
DataHubPoint point;
```

Parameters

point
A DataHubPoint object. The name member must be valid.

Description

This virtual method is invoked on receipt of a point value change.

See Also

onPointEcho
onPointEcho

onPointEcho — a virtual method invoked on receipt of a locally changed point value.

Synopsis

For C++:

```cpp
virtual void onPointEcho(point);
CDataHubPoint& point;
```

For Java and C#:

```java
virtual void onPointEcho(point);
DataHubPoint point;
```

Parameters

`point`

A `DataHubPoint` object. The name member must be valid.

Description

This virtual method is invoked on receipt of a locally changed point value.

See Also

`onPointChange`
**onStatusChange**

onStatusChange — a virtual method invoked on change of status.

**Synopsis**

For C++, Java, and C#:

```cpp
virtual void onStatusChange(prev_state, state);
```

```csharp
DHC_tState prev_state;
DHC_tState state;
```

**Parameters**

- **prev_state**
  The previous state.
- **state**
  The current state.

**Description**

This is a virtual method which is invoked on change of status. See `getCnxState` for a definition of the possible states.

**See Also**

`getCnxState`
onSuccess

onSuccess — a virtual method invoked on receipt of a success message.

Synopsis

For C++:

```cpp
virtual void onSuccess(cmd, parms);
```

```cpp
LPCTSTR cmd;
LPCTSTR parms;
```

For Java and C#:

```java
virtual void onSuccess(cmd, parms);
```

```java
String cmd;
String parms;
```

Parameters

- **cmd**
  
  The original command sent to the DataHub to which this reply corresponds.

- **parms**
  
  The list of parameters sent as part of the command, as a single space-separated string, or **null** if none are returned. If the command involved multiple parameters, the list may be truncated.

Description

This virtual method is invoked on receipt of a success message. When a client sends a command to the DataHub, the DataHub will respond with one of:

- information appropriate to the command
- a success message
- a failure message

If a command succeeds, the DataHub will either respond with information appropriate to the command, or it will send a success message if no information should be returned. If a command fails, the DataHub will always respond with a failure message.

A client has the choice of whether to receive the success messages. Commonly you don’t want to expend the bandwidth by receiving success messages for every message you send to the DataHub. A client can turn on and off the success messages by emitting the...
DataHub command \(\text{acksuccess } 0\mid 1\). This command does not have an associated API function wrapper, so you have to emit it using the \texttt{sendLispMessage} command, like this:

```java
connection.sendLispMessage (true, "(acksuccess 1)");
```

The exact syntax of the call depends on whether you are using C++, .Net or Java.

See Also

\texttt{onStatusChange, onAsyncMessage, onError, onConnectionSuccess, onConnectionFailure}
DataHubPoint Methods

These are the methods associated with the DataHubPoint class, listed alphabetically. To see the same methods grouped according to how they are used, please refer to the Categorized List of Methods.
DataHubPoint

DataHubPoint — constructs a DataHubPoint object.

Synopsis

For C++:

```cpp
CDataHubPoint();
void;
CDataHubPoint(point);
const CDataHubPoint& point;
CDataHubPoint(sname);
LPCTSTR sname;
CDataHubPoint(sname, itype, svalue, iconf, iquality, isecurity, ilocked, isseconds, inanoseconds, iflags);
LPCTSTR sname;
int itype;
LPCTSTR svalue;
int iconf;
int iquality;
int isecurity;
int ilocked;
int isseconds;
int inanoseconds;
int iflags;
```

For Java, and C#:

```java
DataHubPoint(sname);
String sname;
DataHubPoint(sname, itype, svalue, iconf, iquality, isecurity, ilocked, isseconds, inanoseconds, iflags);
String sname;
int itype;
String svalue;
int iconf;
int iquality;
```
Parameters

point
An existing point to be copied.

sname
The point name. The ',' character may be used to separate name fields. A point name may be qualified by its corresponding domain as domain:name, otherwise it is treated by the DataHub as belonging to the current default domain. Point names must be matched exactly, i.e., the qualified and unqualified point names are not the same.

itype
The type of the point: PT_TYPE_STRING, PT_TYPE_INT32 or PT_TYPE_REAL.

svalue
A string representing the value, to be converted according to the specified type.

iconf
The point confidence, typically 0-100.

iquality
The point quality, typically PT_QUALITY_GOOD or PT_QUALITY_BAD.

isecurity
The point security.

ilocked
The locked flag for the point.

iseconds
The seconds component of the initial timestamp (since Jan. 1, 1970).

inanoseconds
The nanosecond component of the initial timestamp.

iflags
The point flags.

Description

Constructs a new DataHubPoint object.

See Also

DataHubPoint Class, Categorized List of Methods
~DataHubPoint

~DataHubPoint — destroys a DataHubPoint object.

Synopsis

For C++:

    ~DataHubPoint ();

    void;

For Java, and C#:

None.

Description

Destroys a DataHubPoint object.

See Also

DataHubPoint Class, Categorized List of Methods
**operator=**

operator= — assigns a new value to a DataHubPoint object (C++ only).

**Synopsis**

**For C++:**

```cpp
CDataHubPoint& operator=(point);
```

```cpp
CDataHubPoint& point;
```

**For Java, and C#:**

None.

**Parameters**

*point*

A DataHubPoint object to be copied into this DataHubPoint object.

**Description**

Copies the name, value and information from the source object. Existing data in the destination object is cleared. A memory exception may occur due to allocation of the name and string type data.

**See Also**

DataHubPoint Class, Categorized List of Methods, clear
clear

clear — clears the point.

Synopsis

For C++, Java, and C#:

```c
void clear();
void;
```

Description

Clears the point, releasing the current name and string values, if any. The cleared point is a valid point of type PT_TYPE_INT32, with value of 0 and PT_QUALITY_BAD.
**Synopsis**

For **C++**:

```cpp
void copy(point);
const CDataHubPoint& point;
```

For **Java, and C#**:

```java
void copy(point);
DataHubPoint point;
```

**Parameters**

`point`

A `DataHubPoint` object to be copied into this `DataHubPoint` object.

**Description**

Please refer to the `operator=` method.
getConfidence

getConfidence — retrieves the user's point confidence.

Synopsis

For C++, Java, and C#:

```c
int  getConfidence();
void;
```

Returns

The confidence value of this point object.
getDateString

dateString — generates a 'standard' timestamp data/time representation.

Synopsis

For C++:

CString getDateString();

void;

For C#:

String getDateString();

void;

Returns

A String object representing the current timestamp in exactly 26 characters as Day Mmm dt hr:mn:sc yyyy. For example Wed Jan 02 02:03:55 1980.

Description

A simple convenience function to generate a timestamp string.
getDoubleValue

getDoubleValue — retrieves the point data as a double-typed value.

Synopsis

For C++, Java, and C#:

```cpp
double getDoubleValue();
void;
```

Returns

The current object value as a double.

Description

This method will convert the point value type if needed. If the string cannot be converted, 0.0 is returned. Strings representing hex, octal, or binary integers generate 0.0.
getFlags

getFlags — retrieves the flags property of the point.

Synopsis

For C++, Java, and C#:

```c
int getFlags();
void;
```

Returns

The current point flags.
getIntValue

getIntValue — retrieves the point data as an int-typed value.

Synopsis

For C++, Java, and C#:

```cpp
int getIntValue();

void;
```

Returns

The current object value as an integer.

Description

This method will convert the point value type if needed. If the string cannot be converted, 0 is returned. Strings representing hex, octal, or binary integers generate 0.
getListeners

getListeners — retrieves listeners on the point (Java only).

Synopsis

For Java only:

```java
LinkedList getListeners();
void;
```

Returns

A list of the current listeners on the point.
getLocked

getLocked — retrieves the locked property of the point.

Synopsis

For C++ and C#:

```cpp
bool getLocked();
void;
```

For Java:

```java
boolean getLocked();
void;
```

Returns

The current locked status of the point.
getName

getName — retrieves the point name.

Synopsis

For C++:

```cpp
CString getName();
void;
```

For Java, and C#:

```java
String getName();
void;
```

Returns

The point name.
getNanoseconds

getNanoseconds — retrieves the timestamp nanoseconds component.

Synopsis

For C++, Java, and C#:

```c
int getNanoseconds();
void;
```

Returns

The nanoseconds component of the timestamp.
getQuality

getQuality — retrieves the point quality.

Synopsis

For C++, Java, and C#:

```cpp
int getQuality();

void;
```

Returns

The point quality.
getQualityString

getQualityString — generates a string representing the point quality.

Synopsis

For C++:

    CString getQualityString();
    void;

For Java, and C#:

    String getQualityString();
    void;

Returns

    A string representing the quality assigned to the point.
**getSeconds**

getSeconds — retrieves the timestamp seconds component.

**Synopsis**

For C++, Java, and C#:

```c
int getSeconds();
void;  
```

**Returns**

The timestamp seconds component.
getSecurity

getSecurity — retrieves the security property of the point.

Synopsis

For C++, Java, and C#:

```cpp
int getSecurity();
void;
```

Returns

The point security.
**getStringValue**

getStringValue — retrieves the point data as a string.

**Synopsis**

For C++:

```cpp
CString getStringValue();
void;
```

For Java, and C#:

```java
String getStringValue();
void;
```

**Returns**

A string representing the current point value.

**Description**

This method will convert the point value type if needed. Values of type `real` are converted using the `%g` format.
**getype**

*getype* — retrieves the point data type.

**Synopsis**

For C++, Java, and C#:

```cpp
int getType();
void;
```

**Returns**

The point type, one of `PT_TYPE_STRING`, `PT_TYPE_REAL`, or `PT_TYPE_INT32`. 
getUserdata

getUserdata — retrieves the user object associated with the point.

Synopsis

For C++, Java, and C#:

```c
void* getUserdata();
void;
```

Returns

The user data previously set by `setUserData`. 
**qualifyName**

*.qualifyName* — creates a point name string qualified by a domain name.

**Synopsis**

**For C++:**

```c
static CString qualifyName(domainname, pointname);
```

```c
LPCTSTR domainname;
LPCTSTR pointname;
```

**For Java and C#:**

```c
static String qualifyName(domainname, pointname);
```

```c
String domainname;
String pointname;
```

**Parameters**

`pointname`

The name of the point.

`domainname`

The name of the domain containing the point.

**Returns**

A string with the correctly formatted qualified point name.

**Description**

This utility method is used to add the domain name qualifier to a point name, using the format *domainname:pointname*. If the point name is already qualified, then the domain name is replaced with the new *domainname* specified.

If `setDefaultDomain` is used, and the point belongs to that domain, then it is not necessary to qualify the point. A qualified point name is normally used to reference points from domains other than the application’s default domain.

**Examples**

All of these would reference the same point:

1. `writePoint(_T("domain1:point1"), 1);`
2. `writePoint(CDataHubPoint::qualifyName(_T("domain1"),_T("point1")), 1);`
3. `setDefaultDomain(_T("domain1"));
   writePoint(_T("point1"), 1);`

See Also

`setDefaultDomain, unqualifyName, writePoint, registerPoint`
removeListener

removeListener — removes a listener from the point (Java only).

Synopsis

For Java only:

```java
void removeListener(listener);
DataHubListener listener;
```

Parameters

`listener`  
Not yet documented.

Returns

Not yet documented.
setConfidence

setConfidence — sets the user's confidence in a point.

Synopsis

For C++, Java, and C#:

```c
void setConfidence(intconf);
```

```c
inticonf;
```

Parameters

`iconf`

The point confidence, typically 0-100.

Description

Sets the user's confidence in a point. See DataHubPoint.
setInfo

setInfo — sets the information properties of a point.

Synopsis

For C++:

```cpp
void setInfo(point);

CDataHubPoint& point;

void setInfo(iconf, iquality, isecurity, ilocked, iseconds, inanoseconds, iflags);

int iconf;
int iquality;
int isecurity;
int ilocked;
int iseconds;
int inanoseconds;
int iflags;
```

For Java, and C#:

```java
void setInfo(point);

DataHubPoint point;

void setInfo(iconf, iquality, isecurity, ilocked, iseconds, inanoseconds, iflags);

int iconf;
int iquality;
int isecurity;
int ilocked;
int iseconds;
int inanoseconds;
int iflags;
```

Parameters

`point`

An existing point from which to copy all the information values (not name, type or value).

`iconf`

The point confidence, typically 0-100.
quality
   The point quality, typically PT_QUALITY_GOOD or PT_QUALITY_BAD.

security
   The point security.

locked
   The locked flag for the point.

seconds
   The seconds component of the initial timestamp (since Jan. 1, 1970).

nanoseconds
   The nanosecond component of the initial timestamp.

flags
   The point flags.

**Description**

Sets all the information properties of a point.
setLocked

setLocked — sets the locked property of the point.

Synopsis

For C++ and C#:

```cpp
void setLocked(ilocked);
bool ilocked;
```

For Java:

```java
void setLocked(ilocked);
boolean ilocked;
```

Parameters

`ilocked`

The locked flag for the point.

Description

Sets the locked property of the point.
**setName**

`setName` — assigns a name to the point.

**Synopsis**

For C++:

```c
void setName(sname);
```

LPCTSTR `sname`;

```c
void setName(sdomain, sname);
```

LPCTSTR `sdomain`;

LPCTSTR `sname`;

For Java, and C#:

```java
void setName(sname);
```

String `sname`;

**Parameters**

- `sname`
  - A string containing the name to be assigned to the point.

- `sdomain`
  - A string containing the domain name.

**Description**

Assigns a name to the point. Any existing name is released. See notes regarding `sname` parameter of `DataHubPoint`. If a domain is specified, then the qualified point name is constructed.
**setQuality**

setQuality — sets the point quality.

**Synopsis**

For C++, Java, and C#:

```cpp
void setQuality(iquality);
int iquality;
```

**Parameters**

- **iquality**
  
  The point quality, typically PT_QUALITY_GOOD or PT_QUALITY_BAD.

**Description**

Sets the point quality.
setSecurity

setSecurity — sets the security property of the point.

Synopsis

For C++, Java, and C#:

```c++
void setSecurity(int isecurity);
```

```java
int isecurity;
```

**Parameters**

- `isecurity`
  - The point security.

**Description**

Sets the security property of the point.
setTimeStamp

setTimeStamp — sets the point timestamp in various ways.

Synopsis

For C++, Java, and C#:

```cpp
void setTimeStamp(seconds, nanoseconds);

int seconds;
int nanoseconds;

void setTimeStamp(datetime);

DATE datetime;

void setTimeStamp();

void;
```

Parameters

- **seconds**
  - The number of seconds since Jan 1, 1970.

- **nanoseconds**
  - The sub-second component of the timestamp, typically accurate to the extent permitted by the OS.

- **datetime**
  - Time expressed in the Microsoft DATE format, the number of days since Dec.30, 1899.

Description

The parameter-less version of this method will set the timestamp to the current OS time.
setFlags

setFlags — sets the flags property of the point.

Synopsis

For C++, Java, and C#:

```cpp
void setFlags(flags);

int flags;
```

Parameters

iflags

The point flags.

Description

Sets the flags property of the point.
**setUserdata**

`setUserdata` — associates the point with a user object.

**Synopsis**

For C++, Java, and C#:

```c
void setUserdata(userdata);
void* userdata;
```

**Parameters**

`userdata`

Any user-supplied data of size (void*).

**Description**

Sets the `userdata` property of a point. The user is free to set this to anything (of the appropriate size). When a point is updated in the `DataHubConnector` point cache, this property is maintained, providing the user with a facility for maintaining an association between a point and an object in the user's application space.

**See Also**

`getUserData`, `initializePointCache`
**setValue**

*setValue* — sets the point data to the specified type and value.

**Synopsis**

For C++:

```cpp
void setValue(point);
CDataHubPoint& point;
void setValue(svalue);
LPCTSTR svalue;
void setValue(dvalue);
double dvalue;
void setValue(ivalue);
int ivalue;
```

For Java, and C#:

```java
void setValue(svalue);
String svalue;
void setValue(dvalue);
double dvalue;
void setValue(ivalue);
int ivalue;
```

**Parameters**

- **point**
  
  A `DataHubPoint` object whose data type and value are to be copied into this `DataHubPoint` object.

- **svalue**
  
  The desired string value.

- **dvalue**
  
  The desired double value.
**i**value

The desired integer value.

**Description**

Sets the object's data value and sets the type accordingly.
**setFromString**

`setFromString` — sets the point data to the value represented by a string.

**Synopsis**

**For C++:**

```cpp
void setFromString(svalue);
LPCTSTR svalue;

void setFromString(svalue, itype);
LPCTSTR svalue;
int itype;
```

**For Java, and C#:**

```java
void setFromString(svalue);
String svalue;

void setFromString(svalue, itype);
String svalue;
int itype;
```

**Parameters**

**sValue**

A string to be used to set the value.

**itype**

The type of the point: `PT_TYPE_STRING`, `PT_TYPE_INT32` or `PT_TYPE_REAL`.

**Description**

If the type is specified, then the string is converted as required. If the type is not specified, then the method determines whether the string represents an integer or double value, converting to that type if possible.
unqualifyName

unqualifyName — removes the domain name qualifier from a point name.

Synopsis

For C++:

```cpp
static CString unqualifyName(pointname);
LPCTSTR pointname;
```

For Java and C#:

```java
static String unqualifyName(pointname);
String pointname;
```

Parameters

pointname

The name of the point.

Returns

The pointname with the domain name qualifier, if any, removed.

Description

This utility method is used to remove a domain name qualifier from a point name. See qualifyName for more information on qualifying a point name with a domain name.

See Also

qualifyName, registerPoint
Gamma
# Table of Contents

Programmers Manual ..................................................................................................... 1
Introduction .................................................................................................................. 2
  What is Gamma? ........................................................................................................ 2
Getting Started .......................................................................................................... 3
  Interactive Mode .................................................................................................... 3
  Executable Programs ......................................................................................... 4
Symbols and Values ...................................................................................................... 5
Basic Data Types and Mechanisms .............................................................................. 8
  Numeric Types ....................................................................................................... 8
    Integer ............................................................................................................... 8
    Real .................................................................................................................. 8
    Fixed-point Real ............................................................................................... 8
    Number Operators .............................................................................................. 8
  Logical Types ......................................................................................................... 9
  Strings .................................................................................................................... 10
  Lists and Arrays .................................................................................................... 10
  Constants ............................................................................................................... 11
  Operators and Expressions .................................................................................... 12
  Comments ............................................................................................................. 12
  Reserved Words ..................................................................................................... 13
Memory Management .................................................................................................. 13
Tutorial I .................................................................................................................... 15
  Lists ....................................................................................................................... 15
  "Hello world" program ....................................................................................... 17
Control Flow ............................................................................................................... 20
  Statements ............................................................................................................ 20
    Conditionals ....................................................................................................... 20
    Loops .................................................................................................................. 21
    Goto, Break, Continue, Return .......................................................................... 21
  Function Calls ....................................................................................................... 22
Event Handling .......................................................................................................... 22
  Interprocess Communication Message Events ..................................................... 23
  Timers .................................................................................................................... 23
  Symbol Value Events (Active Values) ................................................................... 25
  Cogent DataHub Point Events (Exception Handlers) ........................................... 27
  Windowing System Events .................................................................................... 27
  Signals .................................................................................................................... 27
Error Handling ............................................................................................................ 28
  Situations that might cause Gamma to crash ....................................................... 28
Tutorial II .................................................................................................................... 30
  Error Handling - try/catch, protect/unwind ......................................................... 30
  Dynamic Scoping .................................................................................................. 32
  Error Handling - interactive session ..................................................................... 34
Functions and Program Structure ............................................................................... 37
Function Definition ................................................................. 37
Function Arguments ........................................................................... 38
  Variable number of arguments ..................................................... 38
  Optional arguments ..................................................................... 38
  Protection from evaluation .......................................................... 38
  Variable, optional, unevaluated arguments .................................. 39
  Examples .................................................................................. 39
Function Renaming ............................................................................ 41
Loading files ..................................................................................... 42
The main Function ............................................................................. 42
Executable Programs ......................................................................... 43
Running a Gamma Program ............................................................ 43
Command Line Arguments ................................................................ 44
Object Oriented Programming .......................................................... 46
Classes and Instances ................................................................. 46
  Instances ................................................................................ 46
  Methods .................................................................................. 47
Inheritance ....................................................................................... 49
Instance Variables ........................................................................... 52
Class Variables ................................................................................ 52
Constructors and Destructors ......................................................... 53
Polymorphism .................................................................................... 55
  Operator Overloading ................................................................. 55
Binary Classes and User Classes ....................................................... 56
Tutorial III ........................................................................................ 58
Classes and OOP .............................................................................. 58
Interactive Development and Debugging ......................................... 62
Interactive Mode Implementation .................................................... 62
Getting On-Line Help for Functions ................................................. 62
Examining Variables in a Class or Instance ....................................... 63
Using the Debug Prompt ................................................................. 64
Debugging a program ........................................................................ 64
  Interacting with an Active Program .............................................. 64
  Trapping and Reporting Errors .................................................. 66
  Determining Error Location ....................................................... 67
  Filtering Object Query Output .................................................... 67
Advanced Types and Mechanisms ................................................... 69
Symbols ........................................................................................... 69
  Undefined symbols ..................................................................... 69
  Uniqueness of Symbols ............................................................. 69
  Properties ................................................................................ 71
  Predefined Symbols .................................................................. 71
Evaluation ........................................................................................ 71
  Evaluation of a Symbol ............................................................. 71
  Evaluation of a List ................................................................. 71
  Evaluation to Itself ................................................................. 72
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literal Syntax and Evaluation</td>
<td>72</td>
</tr>
<tr>
<td>Literal Expressions</td>
<td>73</td>
</tr>
<tr>
<td>Deferring Expression Evaluation</td>
<td>73</td>
</tr>
<tr>
<td>Literal Function Arguments</td>
<td>74</td>
</tr>
<tr>
<td>Partially Evaluated Literal</td>
<td>75</td>
</tr>
<tr>
<td>Constructing Variable Names at Run-time</td>
<td>76</td>
</tr>
<tr>
<td>Literal Array Syntax</td>
<td>76</td>
</tr>
<tr>
<td>Input and Output</td>
<td>78</td>
</tr>
<tr>
<td>Referencing Files</td>
<td>78</td>
</tr>
<tr>
<td>Lisp and Gamma I/O mechanisms</td>
<td>78</td>
</tr>
<tr>
<td>Writing</td>
<td>79</td>
</tr>
<tr>
<td>Print vs. Princ</td>
<td>79</td>
</tr>
<tr>
<td>Write vs. Writec</td>
<td>79</td>
</tr>
<tr>
<td>Terpri</td>
<td>79</td>
</tr>
<tr>
<td>Pretty Printing</td>
<td>79</td>
</tr>
<tr>
<td>Printing Circular References</td>
<td>80</td>
</tr>
<tr>
<td>Reading</td>
<td>81</td>
</tr>
<tr>
<td>Reading Gamma Expressions</td>
<td>81</td>
</tr>
<tr>
<td>Reading Arbitrary ASCII Data</td>
<td>82</td>
</tr>
<tr>
<td>Reading Binary Data</td>
<td>82</td>
</tr>
<tr>
<td>Special Topics</td>
<td>83</td>
</tr>
<tr>
<td>Modifying QNX Process Environment Variables</td>
<td>83</td>
</tr>
<tr>
<td>QNX 4 Interprocess Communication (IPC)</td>
<td>83</td>
</tr>
<tr>
<td>Cogent IPC</td>
<td>85</td>
</tr>
<tr>
<td>Cogent IPC Service Modules</td>
<td>85</td>
</tr>
<tr>
<td>Cogent IPC Advanced services</td>
<td>86</td>
</tr>
<tr>
<td>Cogent DataHub</td>
<td>92</td>
</tr>
<tr>
<td>Cogent DataHub Exceptions and Echos</td>
<td>93</td>
</tr>
<tr>
<td>A. Function List</td>
<td>95</td>
</tr>
<tr>
<td>B. GNU Lesser General Public License</td>
<td>107</td>
</tr>
<tr>
<td>Gamma Reference</td>
<td>117</td>
</tr>
<tr>
<td>I. Symbols and Literals</td>
<td>118</td>
</tr>
<tr>
<td>Data Types and Predicates</td>
<td>119</td>
</tr>
<tr>
<td>undefined_p, undefined_symbol_p</td>
<td>122</td>
</tr>
<tr>
<td>Literals</td>
<td>124</td>
</tr>
<tr>
<td>Predefined Symbols</td>
<td>127</td>
</tr>
<tr>
<td>Reserved Words</td>
<td>131</td>
</tr>
<tr>
<td>t</td>
<td>132</td>
</tr>
<tr>
<td>nil</td>
<td>133</td>
</tr>
<tr>
<td>gamma, phgamma</td>
<td>134</td>
</tr>
<tr>
<td>II. Operators</td>
<td>137</td>
</tr>
<tr>
<td>Operator Precedence and Associativity</td>
<td>138</td>
</tr>
<tr>
<td>Arithmetic Operators</td>
<td>139</td>
</tr>
<tr>
<td>Assignment Operators</td>
<td>141</td>
</tr>
<tr>
<td>Binary Operator Shorthands</td>
<td>143</td>
</tr>
<tr>
<td>Bitwise Operators</td>
<td>145</td>
</tr>
</tbody>
</table>
IV. Core Functions ............................................................................................. 185
   call .................................................................................................................. 186
   class_add_cvar .............................................................................................. 188
   class_add_ivar ............................................................................................... 190
   class_name ..................................................................................................... 192
   class_of ........................................................................................................... 193
   defclass ........................................................................................................... 194
   defmacro, defmacroe ...................................................................................... 195
   defun, defune, ............................................................................................... 196
   defmethod ....................................................................................................... 197
   defvar .............................................................................................................. 198
   destroy ............................................................................................................ 200
   eq, equal ......................................................................................................... 201
   error ................................................................................................................... 203
   eval ................................................................................................................... 204
   eval_list .......................................................................................................... 206
   eval_string ...................................................................................................... 207
   force, forceq, forceqq .................................................................................. 208
   funcall .............................................................................................................. 210
   function_args .................................................................................................. 211
   function_body ................................................................................................ 213
   function_name ................................................................................................ 214
   getprop ............................................................................................................ 215
   has_cvar ......................................................................................................... 216
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIII. Math</td>
<td>331</td>
</tr>
<tr>
<td>VII. Data Type Conversion</td>
<td>321</td>
</tr>
<tr>
<td>VI. Strings and Buffers</td>
<td>286</td>
</tr>
</tbody>
</table>

### VI. Strings and Buffers
- `bdelete` ........................................................................................................... 287
- `binert` ................................................................................................................ 288
- `buffer` ................................................................................................................. 290
- `buffer_to_string` ................................................................................................. 291
- `format` .................................................................................................................. 292
- `make_buffer` ........................................................................................................ 294
- `open_string` ......................................................................................................... 295
- `parse_string` ....................................................................................................... 296
- `raw_memory` ......................................................................................................... 298
- `shell_match` ........................................................................................................... 299
- `shorten_buffer` .................................................................................................... 301
- `strchr, strstr` ...................................................................................................... 302
- `strcmp, stricmp` ................................................................................................... 303
- `string` .................................................................................................................... 305
- `stringc` ................................................................................................................. 306
- `string_file_buffer` .............................................................................................. 307
- `string_split` ......................................................................................................... 308
- `string_to_buffer` ................................................................................................. 310
- `strcvt` .................................................................................................................... 311
- `strlen` .................................................................................................................... 312
- `strncmp, strnicmp` .............................................................................................. 313
- `strstr` ..................................................................................................................... 314
- `strrev` ..................................................................................................................... 315
- `substr` .................................................................................................................... 316
- `substr` .................................................................................................................... 317
- `tolower` .................................................................................................................. 319
- `toupper` .................................................................................................................. 320

### VII. Data Type Conversion
- `bin` ....................................................................................................................... 322
- `char` ....................................................................................................................... 323
- `char_val` .................................................................................................................. 324
- `dec` ........................................................................................................................ 325
- `hex` ........................................................................................................................ 326
- `int` ........................................................................................................................ 327
- `number` .................................................................................................................... 328
- `oct` ........................................................................................................................ 329
- `symbol` .................................................................................................................... 330

### VIII. Math
- `acos, asin, atan, atan2` ......................................................................................... 332
- `and, not, or` .......................................................................................................... 334
- `band, bnot, bor, bxor` ......................................................................................... 336
- `ceil` ....................................................................................................................... 338
- `cfand, cfor` ........................................................................................................... 339
- `conf, set_conf` ..................................................................................................... 340
IX. Input/Output ................................................................................................ 356
    close ........................................................................................................... 357
    fd_close ................................................................................................. 358
    fd_data_function .................................................................................. 359
    fd_eof_function .................................................................................... 360
    fd_open ..................................................................................................... 361
    fd_read ...................................................................................................... 362
    fd_to_file .................................................................................................. 363
    fd_write ..................................................................................................... 364
    fileno ........................................................................................................ 365
    ioctl ......................................................................................................... 366
    open ........................................................................................................... 367
    pipe .......................................................................................................... 368
    princ, print, pretty_princ, pretty_print ................................................. 369
    pty, ptytio ............................................................................................... 370
    read ............................................................................................................ 371
    read_char, read_double, read_float, read_long, read_short .................. 372
    read_eval_file .......................................................................................... 373
    read_line .................................................................................................... 374
    read_n_chars ........................................................................................... 375
    read_until ............................................................................................... 376
    seek ............................................................................................................ 377
    ser_setup ................................................................................................... 378
    tell ............................................................................................................. 379
    terpri ......................................................................................................... 380
    unread_char ............................................................................................. 381
    write, writec, pretty_write, pretty_writec ............................................ 382
    write_n_chars ........................................................................................... 383

X. File System ................................................................................................. 402
    absolute_path .......................................................................................... 403
    access ....................................................................................................... 404
    basename ................................................................................................. 405
    basename ................................................................................................. 406
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd</td>
<td>407</td>
</tr>
<tr>
<td>chars_waiting</td>
<td>408</td>
</tr>
<tr>
<td>directory</td>
<td>410</td>
</tr>
<tr>
<td>dirname</td>
<td>411</td>
</tr>
<tr>
<td>drain</td>
<td>412</td>
</tr>
<tr>
<td>file_date</td>
<td>414</td>
</tr>
<tr>
<td>file_size</td>
<td>415</td>
</tr>
<tr>
<td>flush</td>
<td>416</td>
</tr>
<tr>
<td>getcwd</td>
<td>418</td>
</tr>
<tr>
<td>is_busy</td>
<td>419</td>
</tr>
<tr>
<td>is_dir</td>
<td>420</td>
</tr>
<tr>
<td>is_file</td>
<td>421</td>
</tr>
<tr>
<td>is_readable</td>
<td>422</td>
</tr>
<tr>
<td>is_writable</td>
<td>423</td>
</tr>
<tr>
<td>mkdir</td>
<td>424</td>
</tr>
<tr>
<td>path_node</td>
<td>426</td>
</tr>
<tr>
<td>rename</td>
<td>427</td>
</tr>
<tr>
<td>root_path</td>
<td>428</td>
</tr>
<tr>
<td>tmpfile</td>
<td>429</td>
</tr>
<tr>
<td>unbuffer_file</td>
<td>430</td>
</tr>
<tr>
<td>unlink</td>
<td>431</td>
</tr>
<tr>
<td>atexit</td>
<td>433</td>
</tr>
<tr>
<td>block_signal</td>
<td>435</td>
</tr>
<tr>
<td>unblock_signal</td>
<td>435</td>
</tr>
<tr>
<td>errno</td>
<td>437</td>
</tr>
<tr>
<td>exec</td>
<td>439</td>
</tr>
<tr>
<td>exit_program</td>
<td>440</td>
</tr>
<tr>
<td>fork</td>
<td>442</td>
</tr>
<tr>
<td>getenv</td>
<td>444</td>
</tr>
<tr>
<td>gethostname</td>
<td>445</td>
</tr>
<tr>
<td>getnid</td>
<td>446</td>
</tr>
<tr>
<td>getpid</td>
<td>447</td>
</tr>
<tr>
<td>getsockopt</td>
<td>448</td>
</tr>
<tr>
<td>setsockopt</td>
<td>448</td>
</tr>
<tr>
<td>kill</td>
<td>451</td>
</tr>
<tr>
<td>nanosleep</td>
<td>453</td>
</tr>
<tr>
<td>setenv</td>
<td>454</td>
</tr>
<tr>
<td>shm_open</td>
<td>455</td>
</tr>
<tr>
<td>shm_unlink</td>
<td>457</td>
</tr>
<tr>
<td>signal</td>
<td>458</td>
</tr>
<tr>
<td>sleep, usleep</td>
<td>461</td>
</tr>
<tr>
<td>strerror</td>
<td>463</td>
</tr>
<tr>
<td>system</td>
<td>464</td>
</tr>
<tr>
<td>tcp_accept</td>
<td>465</td>
</tr>
<tr>
<td>tcp_connect</td>
<td>466</td>
</tr>
<tr>
<td>tcp_listen</td>
<td>467</td>
</tr>
<tr>
<td>wait</td>
<td>468</td>
</tr>
</tbody>
</table>

**XI. OS APIs**

This section enumerates various system calls and APIs related to operating system functionality, including system calls for process management, file handling, signal handling, and network communication.
XII. Dynamic Loading ........................................................................................................... 470
  Autoload ........................................................................................................................ 471
  autoload_undefined_symbol ......................................................................................... 474
  AutoMapFunction ......................................................................................................... 476
  ClearAutoLoad .............................................................................................................. 477
  dclose ........................................................................................................................... 478
  derror ............................................................................................................................ 479
  dlfunc ............................................................................................................................ 480
  DllLoad .......................................................................................................................... 481
  dimethod ....................................................................................................................... 482
  NoAutoLoad .................................................................................................................. 483
  dlopen ............................................................................................................................ 485

XIII. Profiling and Debugging ............................................................................................. 487
  allocated_cells ............................................................................................................. 488
  eval_count .................................................................................................................... 489
  free_cells ...................................................................................................................... 490
  function_calls .............................................................................................................. 491
  function_runtime ......................................................................................................... 493
  gc ..................................................................................................................................... 494
  gc_blocksize ................................................................................................................. 495
  gc_enable ....................................................................................................................... 496
  gc_newblock .................................................................................................................. 497
  gc_trace ......................................................................................................................... 498
  profile ............................................................................................................................ 499
  set_autotrace .............................................................................................................. 501
  set_breakpoint .............................................................................................................. 502
  time .................................................................................................................................. 503
  trace, notrace ............................................................................................................... 504

XIV. Miscellaneous ............................................................................................................. 506
  apropos ........................................................................................................................... 507
  create_state, enter_state, exit_state ............................................................................ 509
  gensym .......................................................................................................................... 511
  modules ........................................................................................................................... 512
  stack ............................................................................................................................... 513

XV. IPC .................................................................................................................................. 515
  add_hook .......................................................................................................................... 516
  close_task ....................................................................................................................... 519
  _destroy_task ............................................................................................................... 520
  init_async_ipc ............................................................................................................... 521
  init_ipc ........................................................................................................................... 522
  isend ................................................................................................................................ 523
  locate_task ..................................................................................................................... 524
  locate_task_id ............................................................................................................... 526
  name_attach ................................................................................................................... 528
  nserve_query .................................................................................................................. 529
  remove_hook .................................................................................................................... 531
run_hooks ........................................................................................................... 533
send ..................................................................................................................... 534
send_async ......................................................................................................... 536
send_string ......................................................................................................... 538
send_string_async .............................................................................................. 539
taskdied, taskstarted ......................................................................................... 541
task_info .............................................................................................................. 543

XVI. Events and Callbacks .................................................................................. 545
add_set_function ................................................................................................. 546
flush_events ........................................................................................................ 548
next_event, next_event_nb .................................................................................. 549
remove_set_function ........................................................................................... 551
when_set_fns ....................................................................................................... 552

XVII. Time, Date, and Timers ............................................................................. 553
after ....................................................................................................................... 554
at ............................................................................................................................ 556
block_timers, unblock_timers .............................................................................. 558
cancel ..................................................................................................................... 559
clock, nanoclock .................................................................................................. 560
date ........................................................................................................................ 561
date_of .................................................................................................................. 562
every ....................................................................................................................... 563
gmtime .................................................................................................................... 565
localtime ............................................................................................................... 567
mktime .................................................................................................................. 569
timer_is_proxy ..................................................................................................... 570

XVIII. Cogent DataHub ...................................................................................... 571
add_exception_function, add_echo_function ..................................................... 572
lock_point .............................................................................................................. 574
point_locked .......................................................................................................... 575
point_nanoseconds ............................................................................................. 576
point_seconds ....................................................................................................... 577
point_security ...................................................................................................... 578
read_existing_point, read_point .......................................................................... 579
register_all_points ............................................................................................... 581
register_exception ............................................................................................... 583
register_point, register_existing_point ............................................................. 584
remove_echo_function ......................................................................................... 586
remove_exception_function ............................................................................... 587
secure_point ......................................................................................................... 588
set_domain ........................................................................................................... 590
set_security .......................................................................................................... 592
unregister_point ................................................................................................... 594
when_echo_fns, when_exception_fns ..................................................................... 596
write_existing_point, write_point ....................................................................... 597
Introduction

What is Gamma?

Gamma is an interpreter, a high-level programming language that has been designed and optimized to reduce the time required for building applications. It has extensions that support HTTP and MySQL.

With Gamma a user can quickly implement algorithms that are far harder to express in other languages such as C. Gamma lets the developer take advantage of many time-saving features such as memory management and improved GUI support. These features, coupled with the ability to fully interact with and debug programs as they run, mean that developers can build, test and refine applications in a shorter time frame than when using other development platforms.

Gamma is an improved and expanded version of our previous Slang Programming Language for QNX and Photon. Gamma was originally available on QNX 4, QNX 6 and Linux, and is now only used with the Cogent DataHub in Microsoft Windows.

The implementation of Gamma is based on a powerful SCADALisp engine. SCADALisp is a dialect of the Lisp programming language which has been optimized for performance and memory usage, and enhanced with a number of internal functions. All references in this manual to Lisp are in fact to the SCADALisp dialect of Lisp.

You could say Gamma’s object language is Lisp, just like Assembler is the object language for C. Knowing Lisp is not a requirement for using Gamma, but it can be helpful. All necessary information on Lisp and how it relates to Gamma is in the Input and Output chapter of this guide.

The syntax of Gamma is very similar to C, so programmers familiar with C can start programming in Gamma almost immediately. Knowledge of pointers and memory allocation is not necessary for using Gamma.
Getting Started

Interactive Mode

You can invoke the Gamma engine by typing

```
[sh]$ gamma
```

at the shell prompt.

It will return the following Gamma prompt:

```
Gamma>
```

Now you can start writing instructions to Gamma and get an immediate response. If you define a variable `a` without assigning it a value, Gamma will respond with the message that the symbol is undefined and suggest debugging:

```
Gamma> a;
Symbol is undefined: a
debug 1>
```

Type `Ctrl - D` to return to the Gamma prompt and assign `a` a value:

```
Gamma> a = 5;
5
Gamma>
```

This time Gamma responds with the value assigned to the variable. In Gamma a variable must be assigned a value. The library function `undefined_p` can be used to test if a variable is defined:

```
Gamma> undefined_p (b);
t
Gamma> b = 1;
1
Gamma> undefined_p (b);
nil
Gamma>
```

This function returns `t` for true and `nil` for false. The objects `t` and `nil` are discussed in more detail in the Logical Types section of the Basic Data Types and Mechanisms chapter.

A function is defined in Gamma using a `function` statement:

```
Gamma> function MyFunc (a) { a *= 10;}
```
(defun MyFunc (a) (*= a 10))
Gamma>

Gamma returns the function definition in Lisp syntax. It reflects the fact that Gamma is using the Lisp engine internally. Basically, Lisp displays functions as lists, surrounded by parentheses. The first word in every Gamma function definition is `defun` because that is the Lisp function for defining functions. After that, the function name is listed, followed by its arguments and code, which is also in Lisp format.

The `MyFunc` function called with 12 as its argument will return the value 120, as follows:

Gamma> `MyFunc` (12);
120
Gamma>

Notice that no type specification is used. An important feature of Gamma is that it is an abstractly typed language, making it unnecessary to specify the type of a data object in order to be able to use it. This does not mean that objects do not have types, but rather the system does not require that the type of an object be known until the code actually executes. This allows a function to return entirely different types, depending on what the calculation produces.

For example, a `MIN` function could be defined as:

```
function min (x, y) { if (x < y) x; else y; }
```

This function will return an integer or floating point number depending on the types of the arguments. The arguments do not need to be the same type. Gamma automatically type-casts them, favoring the smallest possible memory allocation. However, the less-than comparison will fail if both arguments are not numeric types.

Finally, `return` statements are not necessary in Gamma. Looking at the function `MyFunc`, we see it returns 120, the result of multiplying `a` by 10. Any Gamma function that executes successfully always returns a value, which is the result of evaluating the last expression in the function. This return value determines the value and type of the function.

This topic is discussed in greater detail in Function Definitions in the Functions and Program Structure chapter.

**Executable Programs**

There is currently no facility for embedding Gamma source code into a stand-alone Gamma executable. However, it is possible to create a Gamma program which appears to be a stand-alone executable to the user. This is done by using the shell's `#!` directive at the top of the Gamma file. Simply include the line indicating the full path for your Gamma program, for example:
Then change the permissions on the Gamma file to be executable. The Gamma file can now be run directly as if it were a stand-alone executable. The directive must appear as the first line of the program, and it must reference the actual Gamma executable, not a link.

The following example program will print the famous "Hello world" message to the screen:

```gamma
#!/usr/cogent/bin/gamma
princ ("Hello world\n");
```

There is no requirement for a function named `main`. If a function with the name `main` is defined, then Gamma automatically starts to execute that function. Thus, the result of the following program:

```gamma
#!/usr/cogent/bin/gamma
function MyPrint()
{
   princ ("Hello world\n");
}
function main()
{
   MyPrint();
}
MyPrint();
```

is that the function `MyPrint` will be executed twice.

Command line arguments can be passed to the Gamma program using the list variable `argv`. For more details see Tutorial I.

**Symbols and Values**

A symbol is a fundamental notion in Gamma. It is a unique word or name within the program. In this sense all symbols are global in Gamma (although they can have local scope), and all references to a particular symbol will be the exact same Gamma object.

A symbol can contain: lowercase letters, uppercase letters, digits, and underscores. Using a symbol character operator, other characters can be used. A `\` character escapes the next character. A `\s` escapes any and all of the characters in the symbol.
Symbols can have values so they can be used as variables. For the time being it is probably easier to think of a symbol as a variable. (For other uses of symbols see the Advanced Types and Mechanisms chapter of this guide). The same symbol can have different values depending on the part of the program in which the reference to the symbol is made.

For example:

```
Gamma> i = 5;
5
Gamma> function MyLocal (n) { local i; for (i = 1; i < n; i++)
princ("i = ", i, "\n");}
(defun ...)
Gamma> MyLocal (3);
i = 1
i = 2
2
Gamma> i;
5
```

In this example the symbol \( i \) is created and the value of 5 is assigned to it. Then in the definition of function \( \text{MyLocal} \) the value of \( i \) is declared to be \textit{local} to the function (or to have \textit{local scope}). The function prints the local values of \( i \) and returns the result of the last calculation. After \( \text{MyLocal} \) returns, the value of the symbol \( i \) remains 5, as defined at the beginning.

Unlike many procedural languages such as C and Pascal, Gamma uses \textit{dynamic scoping}. That means that if a function defines a local variable, then that variable becomes global to any functions it calls. In the example above, suppose we define \( \text{MyScope} \) in the higher call, or at the global level:

```
Gamma> function MyScope (s) { ..........}
```

Then we modify the function \( \text{MyLocal} \) such that now it calls \( \text{MyScope} \):

```
Gamma> function MyLocalNew (n)
{
    local i;
    for (i = 1; i < n; i++)
    {
        princ("i = ", i, "\n");
        MyScope(i);
    }
}
```

The value of \( i \) used as an argument for \( \text{MyScope} \) will be the value of the local variable most recently declared in the calling sequence: \( \text{MyScope} \ (1), \text{MyScope} \ (2), \text{etc.} \) The
scope of the variable is thus determined at run time by the order in which functions are called.

Dynamic scoping in Gamma is very different from the convention in C known as lexical scoping, where the scope of a variable is determined according to the syntactic position in a program where it is declared.
Basic Data Types and Mechanisms

Numeric Types

A number in Gamma is any integer, real, or fixed-point value.

Integer

An integer is any 32-bit integer number. Integers can be read and written in the following representations:

1. decimal
   
   Example: 45, 129000
2. hexadecimal (start with 0x)
   
   Example: 0xf12
3. octal (start with 0o)
   
   Example: 0o777
4. binary (start with 0b)
   
   Example: 0b101001
5. character (enclosed by ' ')
   
   Example: 's'

Real

A real number is a 64-bit double precision floating point number. It can contain a decimal point and it may end with the letter e followed by a signed exponent.

Examples: 0.1, 235.02013576, 5e-2, 3.74e-7

Fixed-point Real

A fixed-point real number is one that is represented by an integer, where the high 16 bits represent an integer value, and the low 16 bits represent a mantissa. There are very few reasons to work with fixed-point numbers unless floating-point error in numeric comparison is intolerable. Fixed-point numbers are created by any numeric function so long as the value of the symbol _fixed_point_ is non-nil. The default value of _fixed_point_ is nil, so that floating point numbers of type real are created by default.

Number Operators

The following operators can be used with numeric data:

• arithmetic (+, -, *, /, %, div)
Example:

```
Gamma> 2 + 3;
5
```

- logical (!, &&, ||)
  
  Example:

```
Gamma> !nil;  
t
Gamma> !t;    
nil
Gamma> !0;    
nil
Gamma> 2 || nil;
```

Logical Types

In Gamma, there are two objects of logical type: `t` and `nil`. `t` is a logically true value, and `nil` is the ONLY logically false value in Gamma. All other objects are considered to be logically true, including the number zero. This is different from the C language where the number zero is treated as logically false.

The operators !, &&, and || can be used with `t` and `nil`.
Strings

A string is a sequence of characters (whose values range from 0x01 to 0xff), stored in consecutive bytes of memory, terminated by the null character \0. Unlike symbols, strings are not unique within the system. Strings are denoted by enclosing them in double quotation marks (for example: "A string."). A string is created by any of the string functions (particularly string), or by reading a string constant in the form "A string."

The following types of operators can be used with strings:

- comparison (!=, ==)

  Example:

  ```gamma
  Gamma> "cat" != "dog";
  t
  Gamma> "cat" == "dog";
  nil
  ```

- logical (!, &&, ||). Strings have the logical value of true in Gamma.

  Example:

  ```gamma
  Gamma> "cat" && "dog" && nil;
  nil
  Gamma> "cat" || "dog" || nil;
  "cat"
  ```

However, strings are normally manipulated in Gamma using the string functions (see Strings and Buffers in the Reference Manual.)

Lists and Arrays

An array is an ordered collection of elements, each of which can be of a different type. Each element is associated with a fixed numeric index into the array, and can be set or read using the functions aset and aref, or through the use of square brackets [. If an attempt is made to set an array at an index beyond the end of the array, the array will increase in size to the given index, filling any undefined intermediate values with nil. If an attempt is made to read an element from an array beyond the size of the array, then aref will return nil, and no error will be reported.

An array is created by a call to array, or by reading a Lisp expression of the form [ element ...]. A side effect of creating an array using the [...] syntax is that
the array will be effectively static, in the sense that if the array is defined within a function, then it will not be re-created during each call to the function. It would appear to exist within the code space of the function.

A list is a linked group of consecutive elements called cons cells. A cons cell consists of a reference to the list element and a forward pointer to the next cons cell. This organization necessarily means that a list is single-directional, and also means that any cons cell within the list is the head of another list, consisting of that cons cell and all cells after it in the list. The last cons cell in a list normally has a forward pointer of nil.

It is possible to create a list whose last cons cell points to a non-nil object. For example, a single cons cell could have a data value of 1, and a forward pointer to number 2. This would create a dotted list, written as (1 . 2) (note the spaces around the dot). Only the last element in a list can be a dotted cons cell. Thus, it is not possible to create a list (1 . 2 3) or (1 . (2 3)). A dotted cons cell is created by a call to cons, or by reading a Lisp expression of the form (x . y). The form (x . nil) is equivalent to (x).

There are two very important functions used to access the members of a list: car(list) and cdr(list). The car function returns the first element of a list. For example, car(list(3,2,1)) returns 3. The cdr function returns the "tail" of the list, that is, the list without the first element. For example, cdr(list(3,2,1)) returns (2 1). The combinations of these two functions allow access to any element(s) of a list. For example, car(cdr(list(3,2,1))) returns 2, the second element of the list. Normally a shortcut cadr(list) is used for car(cdr(list)).

For more information see car, cdr in the Reference Manual.

A list is created by a call to cons or list, or by reading a Lisp expression of the form (x ...), or by evaluating a Lisp expression of the form '(x ...) or `(x ...).

Both lists and arrays can be traversed using the with statement. For example:

```
Gamma> sum = 0;
0
Gamma> with i in list(1,2,3) do {sum += i;}
nil
Gamma> sum;
6
Gamma>
```

For more information on the with statement, see with in the Reference Manual.

**Constants**

A constant is any symbol that has been defined as such with the assignment operator ::=. Note that since the check is made at run-time, the constant is protected, even if the symbol name is evaluated at run-time.
Example:

```
Gamma> CON ::= 5.4545454;
5.4545454999999997663
Gamma> CON = 7;
Script error during command CON = 7; Assignment to constant symbol: CON
Gamma>
```

There are four pre-defined constants in Gamma:

- **PI**
  The value of pi, approximately 3.14159.

- **E**
  The base of the natural logarithm, approximately 2.71828.

- **INF**
  Floating point positive infinity.

- **NAN**
  Floating point not-a-number.

Please refer to [Literals](#) for more information on **INF** and **NAN**.

### Operators and Expressions

Gamma provides operators for the basic arithmetic calculations: addition, subtraction, multiplication, division and taking the modulus. There is also a group of assignment, bitwise, comparison, increment and decrement, logical, and quote operators. For information on operator precedence and associativity see [Operators](#) in the Reference Manual.

Operators are used with one, two, or three values to create expressions. For instance, $4 + 2$ is an expression whose value is 6. In Gamma, every expression has a value.

Not all expressions contain operators, though. The number 5, for example, is also an expression. Generally speaking, an expression in Gamma is anything that can be evaluated. This includes numbers, symbols that have been assigned values, strings, `t`, `nil`, constants, lists, arrays, and so on.

These are also known as symbolic expressions, a term that has been abbreviated to `s_exp`. Since expressions are often used as arguments for functions, you will come across the parameter `s_exp` in function definitions in the Reference Manual.

An expression can become a statement by adding a semicolon. Thus, $4 + 2;$ is a statement. For more information on statements, see the [Statements](#) section in the Control Flow chapter.

### Comments

There are two ways insert comments into Gamma code. To comment out a line, you can use a double slash (`//`) like this:

```
a = 5;
b = 7;
```
// This assigns the value of c.

c = a + b;

To comment out a block of text, you can put the symbol /* at the beginning, and */ at the end, like this:

\[
a = 5;
b = 7;
\]

 /* This assigns the value of c, which is very important to the future of our project. */

c = a + b;

It is not permitted to put an unmatched double quote mark (" ) into the second type of comment. This is a feature that allows you to comment out a string, to include a comment mark in a string, and even to comment out a string that includes comment characters, like this:

\[
/*
   princ ("/* this is what a comment looks like\n");
    princ (" when extended to multiple lines\n");
    princ ("*/\n");
*/
\]

Reserved Words

In Gamma, certain words are predefined and reserved for system use. No symbols can be defined by the user that are identical to these reserved words.

The reserved words are:

class, collect, do, else, for, function, if, local, method, tcollect, while, with.

For more details see Reserved Words entry in the Reference Manual.

Memory Management

The programmer generally does not need to consider the memory management aspects of programming in Gamma because Gamma handles all memory management requirements of a task internally through a mechanism known as garbage collection. This greatly simplifies programming and eliminates errors associated with dangling pointers, freeing unallocated memory, and array overruns so common in languages such as C. Nevertheless, Gamma provides some functions for examining and invoking the garbage collector.
These may be used to determine run-time memory requirements or to ensure that the garbage is collected at pre-determined times.

For more information on garbage collections and the related functions see \texttt{gc} in the Reference Manual.
Tutorial I

This tutorial contains examples of the basic types and mechanisms of Gamma. The first example shows you how to manipulate lists. A list is a very important data type in Gamma. The understanding of list manipulation is a key notion to many of Gamma's constructions. The second example walks you through the famous "Hello world" program.

Lists

The examples below demonstrate some of the basics of list manipulation. Since Gamma uses the Lisp engine, it inherits the rich set of list functions for which Lisp is known.

```gamma
/* The program starts with the line containing the shell's directive #!. It makes a Gamma program appear to be a stand-alone executable to the user. The directive must appear as the first line and must reference the actual Gamma executable, not a link. */

#!/usr/cogent/bin/gamma

/*
 * Lisp defines the functions 'car', 'cdr', and 'cons' as the basic list manipulation functions. The origins of these names have no meaning on today's computers:
 * CAR - Contents of the Address Register
 * CDR - Contents of the Decrement Register
 * CONS - Construct (OK, this makes sense)
 *
 * We can define alternate names for car and cdr here:
 */

head := car;
tail := cdr;

/*
 * Create some lists.
 */

a = list ("My", "dog", "has", "fleas");
b = list (1, 2, 3, 4, 5, 6);

/* The pound sign "#" in front of a symbol prevents Gamma from evaluating the symbol so the symbol is taken literally. Thus, if x = 5, then the function call list (#x, 1) will create the list (x 1) rather than (5 1). */
```
c = list (#a, #b, #c, #d, #e, #f);
d = list (#d, #h, #b, #i, #g, #e, #c);

/*
 * Take the first component of a list.
 */
e = head (a);
princ ("The first component of ", a, " is ", e, "\n");

/*
 * Take the tail of a list.
 */
e = tail (a);
princ ("\nThe tail of ", a, " is ", e, "\n");

/*
 * Concatenate two lists. Notice that list elements do not need to
 * be the same type.
 */
e = append (a, b);
princ ("\nappending ", b, " to ", a, " gives:\n ", e, "\n");

/*
 * Walk a list and print each element
 */
princ ("\nThe elements of 'a' are:\n");
with i in a do
{
    princ (" ", i, "\n");
}

/*
 * Add an element to the beginning of a list
 */
princ ("\nAdding a zero to ", b, "\n");
b = cons (0, b);
princ ("    Gives: ", b, "\n");

/*
 * Take the first element of the tail of the list (the second element)
* We can use combinations of car and cdr to do this. However, Gamma
* predefines a number of car and cdr combinations to make this easier,
* by inserting multiple 'a's and 'd's between the 'c' and the 'r' in
* the words car and cdr.
* For example, caddr(x) is the same as car(cdr(cdr(x)))
* 
* Gamma defines all one_, two_, and three_letter combinations of
* car and cdr.
*/

princ ("\na is ", a, "\n");
x = car (cdr (a));
princ ("     car(cdr(a)) is: ", x, "\n");
x = cadr (a);
princ ("     cadr(a) is: ", x, "\n");

/*
* Some other interesting list functions...
*/

princ ("\n");
princ ("Union of ", c, " and ", d, " is ",
   union (c, d), "\n");
princ ("Intersection of ", c, " and ", d, " is ",
   intersection (c, d), "\n");
princ ("Difference of ", c, " and ", d, " is ",
   difference (c, d), "\n");
princ ("Difference of ", d, " and ", c, " is ",
   difference (d, c), "\n");

"Hello world" program

This example program prints a personalized "Hello" message a given number of times. It
demonstrates variable naming, function definitions and basic control structures, as well as
command line arguments.

The program also shows the basic similarities between Gamma and the C language, and
some important differences which highlight the power of Gamma.

#!/usr/cogent/bin/gamma

/* This function iterates, saying hello to the given name.
In Gamma, all functions return a value, which is the
result of the last evaluated expression in the function.
In this case, we return the string "ok".
*/
function say_hello (name, n)
{
    local i;

    for (i = 0; i < n; i++)
        princ ("Hello ", name, "\n");
"ok";
}

/* A program may optionally have a function main() declared, in which case Gamma will execute the main() function after all of the program file has been read in. If no main() declared, the programmer must explicitly call a function or enter an event loop at some point while reading the file. Any code which is not a function definition will be automatically run AS THE FILE IS READ. This is useful for providing feedback to the user while loading. */

function main ()
{
    /* Define some locally scoped variables. Notice that Gamma implements abstract data typing, so it is not necessary to declare the variable type. */
    local repeat = 1, my_name = "world";

    /* Access the command line arguments. argv is a list of the command line arguments, as strings, where the first argument is the program name. */

    /* The second argument (cadr(argv)) is my_name, if present. */
    if (cadr(argv))
        my_name = cadr (argv);

    /* The third argument (caddr(argv)) is the number of iterations, if present. */
    if (caddr(argv))
        repeat = number (caddr(argv));
/_now print the message */

result = say_hello (my_name, repeat);
princ (result, "\n");
Control Flow

Gamma provides a variety of mechanisms for control flow. These generally fall into the categories of statements, function calls, event handlers and error handlers. Many of these are dealt with in more detail in other sections of this document. All functions have a detailed entry in the Reference Manual.

Statements

A statement in Gamma is any syntactically complete piece of code that can be independently evaluated, written in statement syntax. There are two kinds of statement syntax, as follows:

; A semi-colon at the end of an expression is used to denote a single, one-line statement.

{} Curly brackets surrounding zero or more expressions or statements create a single statement from them. Multiple statements within the curly brackets can be grouped in any combination, and nested in any number of levels. This statement syntax is also referred to as a compound statement or code block. Each expression or statement is evaluated in order, and the result is the result of the last statement or expression. A code block in Gamma does not create a local scope.

Gamma has a number of built-in statements, several of which are explained below. For a complete list of built-in Gamma statements, see Statements in the Reference Manual.

Conditionals

Gamma contains a single conditional statement: if. The if function evaluates its condition, and if the condition is non-nil, then the first statement after the if is performed. If the condition is nil, the else clause of the statement is executed. It is not mandatory that an if statement have an else clause. In the case of nested if statements, an else clause will always bind with the nearest if statement.

For example:

```gamma
... 
if (var == 1) 
  count1 ++;
else if (var == 2) 
  count2 ++;
else if (var == 3) 
  count3 ++;
...
```

The if statement accepts only a single statement for its true and false clauses. Using the code block statement syntax, it is possible to perform more than one action inside an if.

Example:
Loops

Gamma supports three looping statements: \texttt{for}, \texttt{while} and \texttt{with}. The \texttt{for} loop looks exactly like a \texttt{for} loop in C:

\begin{verbatim}
... 
local i;
for (i = 1; i < N; i++)
{
    body_statements
}
...
\end{verbatim}

A \texttt{while} loop is also exactly like the C \texttt{while} loop, though the \texttt{do/while} variant available in C is not supported in Gamma.

Gamma adds the \texttt{with} loop, which walks a list or an array and executes a body statement once for each element in the list or array. The \texttt{with} loop may be instructed to collect the results of the body statement for each iteration, and return the accumulated results as a new list or array. The iteration variable in a \texttt{with} loop is defined only within the body of the loop. For an example, see the Lists and Arrays section in the Basic Data Types and Mechanisms chapter.

Goto, Break, Continue, Return

Gamma does not contain facilities for non-linear local jumps. Many languages provide one or more wrappers on the \texttt{goto} function, such as \texttt{break} (go to the end of the expression), \texttt{continue} (go to the beginning of the expression) and \texttt{return} (go to the end of the function). These facilities can be used on occasion for clarity, but can commonly act to confuse both the programmer and the reader.

In Gamma, such a facility would be very confusing, as it is not always clear which expression constitutes the scope of a break, continue or return. In addition, the execution speed penalty associated with supporting local jumps in a functional language generally outweighs the benefit in the few cases where a local jump would be convenient. Further, because Gamma is dynamically scoped, a local jump would be defined as a jump that does not cross a scope boundary rather than the more common C definition of a jump that does not cross a function boundary. This distinction greatly reduces the value of, and the need for, a local jump capability.
The most common local jump instruction in procedural languages is `return`. This instruction moves the execution to the bottom of the function and supplies a return value for the function. In Gamma, all functions implicitly return a value which is the result of the last expression to be evaluated within the function body. If no expression is evaluated, then the function returns `nil`. If the programmer wishes to return the value of a symbol, he or she can simply write that symbol, followed by a semicolon, as the last statement to be evaluated in the function.

**Function Calls**

Any Gamma expression which makes a call to a function, such as `tan(3.14159)`, causes a change of program flow, entering a new scope and causing execution to be temporarily diverted into the function being called. In most cases, the function simply returns and flow continues at the expression containing the function call.

If an error occurs during a function call, the function will not return, and execution will continue from the most recent `protect/unwind` or `try/catch` construct (see the Error Handling section of this chapter).

**Event Handling**

An *event* is a change to which the system responds. Events include interprocess communication (message exchanging between processes) events, signals, timer events (execution of a block of code at specified time), Graphical User Interface events (clicking on the screen buttons), and DataHub exceptions (a change in value of a DataHub point).

Gamma provides a generalized event handling capability which processes all these events. A Gamma function that responds to an event is called an *event handler*, or a *callback*. A pair of functions, `next_event` and `next_event_nb`, invoke the event handler, that is, any callback function(s) that have been attached to the event. For example, the simple construct:

```gamma
while ( t)
{
    next_event ( );
}
```

placed at the end of the file, together with the set of callback functions defined in the application to handle all the required events, creates a purely event-driven Gamma program. However, unlike typical event-driven systems applications (such as many GUI-based applications), the user has here the opportunity to further process the events, providing a conditional which is more complex than an infinite loop, or even choosing not to receive events. The events will not be processed (or callbacks invoked) until one of the `next_event` calls are made.

The result of `next_event` is the result of executing the callbacks, or `nil` if no event handler has been defined (that is, no callbacks have been attached). The result of `nex-
t_event_nb (nb stands for non-blocking) is the same except that nil is also returned if no event was available.

Attaching callbacks and receiving events depends on the type of event. For example, in Photon the function PtAttachCallback is used to attach actions to clicking on buttons. Normally one does not wait for an explicit event type, that is, the next_event call will process ANY event defined within the system. The following sections describe how some common event types are handled.

Interprocess Communication Message Events

Gamma uses a send/receive/reply (SRR) mechanism to provide interprocess communication via messages. In this context, a message is any valid Gamma or Lisp expression passed synchronously from one process to another, using the send function. The engine treats the message as a null-terminated character string in Lisp syntax (Gamma’s internal representation), which it parses and evaluates. Any expression may be transmitted in this way, including function definitions, function calls, variable names and complex blocks of code. The reply returned is the result of the evaluation.

This process is transparent to the application. The Gamma engine evaluates the incoming message inside an implicit try/catch block to ensure that externally originated expressions cannot accidentally affect the running program. Any errors that occur while the message is being evaluated will be indicated in the return value for the message, but the overall running status of the engine will not be otherwise affected.

For more information see the Interprocess Communication section in the Special Topics chapter.

Timers

A timer is a Gamma expression that is submitted for evaluation at specified time in the future. Timer related events are set up through the every and after functions which provide a relative time delay, and the at function which accepts an absolute time. These functions accept timing parameters and a block of code that will be evaluated when the timer expires. The code can be simply a function name, or can be an entire expression to be evaluated.

A timer will only be handled during a call to next_event, next_event_nb or flush_events. If a timer expires while another operation is being performed, the engine will evaluate the timer code at the next event handling instruction.

In Gamma, by default, timers are internally handled by using proxies. A proxy is non-blocking system message that does not require a reply. Gamma can act on a proxy immediately, or delay a little while in order to finish what it is currently doing. It is this ‘little while' that becomes the limit of the accuracy of the timers in Gamma when they are driven from proxies. You see, most programs written in Gamma are run through an event loop. The maximum time a proxy-based timer can be delayed is the execution time of the longest path of code attached to an event. Since this quantity is totally dependent on how your
code is written and the speed of your CPU, it is difficult to put an exact number on the practical accuracy of Gamma’s proxy-based timers. The maximum resolution of the timers is equal to the OS tick size setting.

**Setting a timer**

A timer is created when any of the following functions are called:

The *after* timer is used to evaluate a piece of code after a given amount of time. It is a one-shot timer that "goes away" after being fired.

The *every* timer is used to evaluate a piece of code at a specified interval.

The *at* timer evaluates the associated code when the current time matches the profile created by six arguments to the function.

For syntax and examples see the Reference Manual.

**Canceling a Timer**

All timer functions return a timer_id that can be used to cancel the timer. To do so, the cancel function is called, using the timer-id for its argument.

**The TIMERS variable:**

On startup of Gamma a variable called TIMERS is initialized. This variable is an array of all the timers currently in the system. Initially, its value is [] (an empty array) but as timers are added to the system this array grows.

```
Gamma> TIMERS;
[]
Gamma> a = every(3,#princ("Hello\n")); 1
Gamma> >TIMERS;
[[860700531 176809668 3 ((princ "Hello\n") 1)]
```

The TIMERS variable was initially an empty array. Once the first timer was added the TIMERS variable contained information on the first timer. The contents of each element can be summarized as:

```
second nanosecond repeat function timer_id
```

The second is the number of seconds since Jan. 1, 1970 which is compatible with the clock and date functions. The nanosecond is the fraction of the second. Combining these two times gives an accurate time that identifies the next time the timer will fire. The repeat is nil if the timer is a once-only "after" timer or an "at" timer. Otherwise, this is the period of an "every" timer. The next item is the code that is evaluated when the timer fires. The last item in this sub-array is the timer-id.
This array grows as timers are added to the system:

```
Gamma> b = every(5, #princ("Test\n");
2
Gamma> _timers_;
[[860700531 176809668 3 ((princ "Hello\n") 1]
[860700563 494732737 5 ((princ "Test\n") 2]]
Gamma>
```

An important point to remember is that the TIMERS variable is an array, and therefore can be referenced as such. To reference the first element in the TIMERS array, use TIMERS [0] just like a regular array reference. Altering the TIMERS array can cause unpredictable behavior, and should be avoided.

**Blocking timers from firing**

Sometimes a segment of code is written which must be executed non-stop, without an interruption from timers. To accomplish this, the code is wrapped between block_timers and unblock_timers functions. In this way the code will be safe from interruption from timers. This blocking is not necessary unless timers have been bound to signals rather than proxies.

**timer_is_proxy function**

This function controls how timers are fundamentally handled within Gamma. By default, timers are handled by the processing of proxies. This allows Gamma to delay the timer, if necessary, when a critical system process is occurring.

Calling the timer_is_proxy function with nil makes all timers operate by using signals. In the QNX 4 operating system, for example, SIGALRM is used, and the attached code is run as a handled signal. Running timers via signals has some very dramatic consequences. First and foremost, when running in this mode all timer code must be signal safe. This status must be analyzed with caution, as most code is not signal safe. This mode should be avoided except in very rare circumstances.

**Symbol Value Events (Active Values)**

Gamma has the capability to generate an event when a variable is modified in any way. A variable that can trigger an event is called an active value. The code that is executed when the variable changes is called the set expression that is effectively a callback.

The add_set_function permits attaching an expression to any defined variable. (It is an error to attach a set expression to a constant symbol.) When the value of a symbol changes, either by being declared within a sub-scope or by being explicitly changed using =, :=, -=, or +=, the Gamma engine checks the symbol for the existence of an associated set expression. If a set expression exists, then it is executed directly after the value of the symbol is changed. This set expression can be any valid Gamma code, and is evaluated
within its own sub-scope. The symbols `value`, `previous` and `this` are defined within this sub-scope to be the current value of the symbol, the previous value of the symbol, and the symbol itself respectively. If an active value causes another active value to change, then that new active value’s set expression is also evaluated. This provides a very simple and powerful means by which forward chaining algorithms such as those in expert systems can be implemented.

Active values provide a very powerful way of constructing an event-driven application with a high degree of cohesiveness. Often, there is some functionality that is related to a derived variable, not an event itself. The function can be attached to that internal variable, decoupling it from the event, and generating clearer and more concise code. In the following example, we wish to attach an action to an alarm condition, which in turn is generated by a change in a measured variable (e.g., the temperature). The `#` sign protects an expression from evaluation, causing it to be treated as a literal. (We discuss Deferring Expression Evaluation in more details later, in the chapter on Advanced Types and Mechanisms).

```
Gamma> temp = 35;
35
Gamma> hi_limit = 40;
40
Gamma> function check_temp (tp)
    {if (tp > hi_limit) alarm_hi_on = t;
     else alarm_hi_on = nil;}
(defun check_temp (tp)
    (if (> tp hi_limit)
        (setq alarm_hi_on t)
        (setq alarm_hi_on nil)))
Gamma> function print_alarm ()
    {if (alarm_hi_on == t) princ("Alarm is on: Temp is over ",
                                hi_limit, ".\n");
     if (alarm_hi_on == nil) princ("Alarm is off.
");}
(defun print_alarm ()
    (progn (if (equal alarm_hi_on t)
        (princ "Alarm is on: Temp is over " hi_limit ".\n")
        (if (equal alarm_hi_on nil)
            (princ "Alarm is off.
"))))
Gamma> add_set_function(#temp,#check_temp(temp));
(check_temp temp)
Gamma> add_set_function(#alarm_hi_on, #print_alarm());
(print_alarm)
Gamma> temp = 38;
Alarm is off.
38
Gamma> temp = 39;
39
Gamma> temp = 42;
Alarm is on: Temp is over 40.
```
Cogent DataHub Point Events (Exception Handlers)

The Cogent DataHub is a data collection and distribution center designed for easy integration with Gamma. A point is a name for data in the Cogent DataHub. An exception handler is a Gamma expression that is attached to a point. The Cogent DataHub can asynchronously transmit any number of point values to a Gamma program, which will then automatically update the value of a symbol named the same as the DataHub point.

The `add_exception_function` and `add_echo_function` functions permit an application to bind an expression to a DataHub point exception in a similar way to `add_set_function` above. If an application can both write a point and receive exceptions from that point, then the DataHub will "echo" the exception originated by the application. The two functions make it possible to distinguish between these two situations. Only the originating task will see a point exception as an echo, while all other tasks will see a normal exception.

In addition, the programmer may attach any number of expressions to the symbol, to be evaluated when the DataHub point changes. The expressions are evaluated within a sub-scope, with the special symbols: `value`, `previous` and `this` defined to be: the current value of the point, the previous value of the point and the point name itself as a symbol. An exception handler is only triggered during a call to `next_event`, `next_event_nb` or `flush_events`.

Windowing System Events

Gamma’s GUI support offers `PtAttachCallback` for Photon and `AttachCallback` for X Windows that permit attaching callbacks to any GUI event. Like the other event handling functions, the user can in fact bind any Gamma expression for execution when the event occurs.

GUI Event Handlers (Callbacks)

A GUI event handler, also known as a `callback`, is an arbitrary expression attached to a particular callback of a widget. A callback may occur whenever a call is made to `next_event`, `next_event_nb`, and `flush_events`. If the appropriate GUI event has occurred, then the Gamma engine automatically evaluates any callback handlers that deal with the event attached to the particular widget. This results in essentially asynchronous program flow, where the callback may occur, from the user’s perspective, at any time during the program execution. In reality, the GUI event is only handled if the system has been instructed to deal with one or more incoming events.

Signals

Signals are the traditional method of asynchronous communication between tasks, in which no data is transferred. A `signal handler` is an expression attached to an operating
system signal, which is delivered asynchronously to the running program. A signal handler is attached by a call to the signal function.

A signal pre-empts any activity except garbage collection, and causes control flow to enter the signal handler. The signal handler should not call non-reentrant functions. It is safe for a signal handler to make a call to the error function, which will throw flow control to the nearest error handler.

Gamma supports the following signals:

```
SIGABRT, SIGBUS, SIGCHLD, SIGCONT, SIGDEV,
SIGEMT, SIGFPE, SIGHUP, SIGILL, SIGINT,
SIGIO, SIGIOT, SIGKILL, SIGNAL_HANDLERS,
SIGPIPE, SIGPOLL, SIGPWR, SIGQUIT, SIGSEGV,
SIGSTOP, SIGSYS, SIGTERM, SIGTRAP, SIGTSTP,
SIGTTIN, SIGTTOU, SIGURG, SIGUSR1, SIGUSR2,
SIGWINCH
```

For the description of signal values see the signal entry in the Reference Manual.

**block_signal & unblock_signal**

There are times when certain portions of code must not be interrupted by certain or all signals. Use the block_signal and unblock_signal functions to protect a process.

**Error Handling**

An error handler is a function that responds to an error. In general, when a program executes, the flow of control moves from one expression to the next, possibly passing into and out of function calls, and following branches and loops as necessary. If an error occurs, however, the flow of control will not move to the next expression, but will jump immediately to the most recently declared error handler, either through the protect/unwind or the try/catch constructs. These constructs are the two pairs of functions available in Gamma which allow for trapping and handling errors. See Tutorial II for more details.

The combination of signal handlers and error handlers can cause a program to jump to a predefined point at any time during its execution. An error can be explicitly caused by a call to the error function.

**Situations that might cause Gamma to crash**

Gamma is a very robust language, particularly in comparison to programming in C. However, the power and ease of use can sometimes lead a programmer to create situations that could crash the Gamma engine. Generally these are errors that would certainly have crashed a C program, and would be considered part of the debug cycle. The following list highlights some situations where care must be taken:
1. A call to `init_ipc` inside a timer or signal handling routine. You should call
   (`init_ipc`) once at the beginning of your program if at all.

2. Abuse of Photon widget resources. While care has been taken to minimize the risk
   of a crash by abusing the Photon widgets, the bottom line is that widgets are raw C
   structures with fairly complex manipulation functions. If you write a bad value into a C
   structure, your program will crash. If you fail to call `PtInit`, your program will crash.
   If you create a non-window widget with no parent, your program will crash. These are
   just facts of life.

3. Gamma provides a ‘wrapper’ for most of the standard C library functions that makes
   the corresponding C function call after extracting the Gamma arguments. If the argu-
   ments passed cause the C function to crash, then your program and the Gamma en-
   gine will crash as well.

Having said these things, we are always interested in hearing about new ways that we can
make Gamma more robust. Please don't hesitate to let us know if you find a weak spot.
Tutorial II

This tutorial includes examples related to control flow in Gamma: namely, error handling and dynamic scoping.

Error Handling - try/catch, protect/unwind

This example demonstrates the error handling mechanisms available in Gamma. There are two basic means of trapping and handling errors.

1. Execute a protected block of code, and specify an error handler which is only executed if an error occurs within the protected code. If an error occurs, the error handling code is executed, and the error condition is cleared. This is the try/catch mechanism.

2. Execute a protected block of code, and specify a second block of code which must be executed even if an error occurs in the first block. Normally when an error occurs, the execution stack is unwound to the nearest error handler, aborting any intervening execution immediately. If a block of code must be run, even when an error occurs, we want to unwind protect that code. After the error is dealt with, it is passed on up the stack rather than being cleared. This is the protect/unwind mechanism.

The code for the example is shown below.

```gamma
#!/usr/cogent/bin/gamma

/*
 * Create a function which has an error in it.
 * The symbol zero is not defined.
 */

function sign (x)
{
    if (x < zero)
        princ ("Negative\n");
    else
        princ ("Positive\n");
}

/*
 * Create a function which checks the sign of a number,
 * but ensures that an error will not terminate the program.
 */

function checkit (x)
{
    princ ("\nEntering a TRY, CATCH block...\n");
    try
    {
```
```c
/* Create a function which checks the sign of a number, 
* and which will print a status message whether or not an error
* occurs, before passing a possible error condition up the stack. 
* The 'princ' statement in this case will always be run, even if 
* an error occurs. 
*/

function unwindit (x)
{
    princ ("
Entering a PROTECT, UNWIND block...
\n"); 
    protect 
    {
        sign (x);
    }
    unwind 
    {
        princ ("Finished checking the sign\n");
    }
}

/*
* Attempt to call this function, but if we get an error, 
* simply print the error message and continue. 
*/
checkit (-5);

/*
* Run the same code, but with zero defined 
*/
zero = 0;
checkit (-5);

/*
* Run the unwind protected function. This should print its unwind 
* message. 
*/
Dynamic Scoping

This example uses the error handling mechanisms from the previous Error Handling section to demonstrate dynamic scoping. Most compiled languages use lexical scoping, which means that a variable is defined only where it is visibly declared, either as an external global, file global, or local variable. Gamma uses dynamic scoping, meaning that a variable is defined in any function which defines it, and in any function which the defining function subsequently calls. This powerful mechanism allows the programmer to override global variables by defining them in a higher scope, and then calling a function which believes itself to be using a global variable.

One useful side-effect of dynamic scoping is that functions and variables do not have to be declared before they are used in other functions. The function or variable only has to be declared when the other function is actually run.

The code for the example is shown below.

```gamma
#!/usr/cogent/bin/gamma

/*
 * Create a function which has an error in it.
 * The symbol zero is not defined.
 */

function sign (x)
{
    if (x < zero)
        princ("Negative\n");
    else
        princ("Positive\n");
}
/*
 * Create a function which checks the sign of a number,
 * but ensures that an error will not terminate the program.
 */

function checkit (x)
{
    try
    {
        sign (x);
    }
    catch
    {
        princ ("Oops: ", _last_error_, "\n");
    }
}

/*
 * Create a function which locally declares the variable 'zero', and
 * then calls the checkit function. Since 'zero' is a local variable,
 * the local value will override the current global definition, which
 * is undefined.
 */

function zero_check (x)
{
    local zero = 0;
    checkit (x);
}

/*
 * Create a function which sets zero to -10, and calls the checkit
 * function.
 */

function minus_ten_check (x)
{
    local zero = -10;
    checkit (x);
}

/*
 * Attempt to call the checkit function with zero not defined.
 */
tutorial II

princ ("With 'zero' undefined...
")
checkit (-5);

/*
 * Now let zero be defined and try again.
 */
princ ("\nWith 'zero' locally defined to 0...
"
);
zero_check (-5);

/*
 * Now run with zero defined as -10
 */
princ ("\nWith 'zero' locally defined to -10...
"
);
minus_ten_check (-5);

/*
 * Finally, try running checkit again from the global scope. Note that
 * zero is undefined once again.
 */
princ ("\nOnce again from the global scope...
"
);
checkit (-5);

error handling - interactive session

The code below provides an example of starting an interactive session in case of an error. In this example a window with two buttons is created. Pressing the button labeled good button will print a message to stdout. Pressing the button labeled error button will cause the UNDEFINED symbol g to be evaluated. This causes an error and starts the interactive session from the catch block. The function that runs the interactive session is designed to be recursive and relies on the dynamic scoping of variables in Gamma. Notice that Lisp grammar is being used in this interactive session. You can query the value of any variable simply by typing the variable name in. For example:

win
but1
but2
(@ win title)

or use some functions like:

(stack)
(* 8 3)
Before terminating the interactive session, try to resize the window. Notice that it does not work because the event loop is temporarily suspended. Now exit the interactive session by typing Ctrl-D and notice that the window can now be resized. Also notice that once the event loop is re-started, the contents of the window are not updated but the callbacks are still active. This happens because Photon was interrupted in the middle of a function call and an error condition now exists between Gamma and the Photon library.

```lisp
#!/usr/cogent/bin/gamma

require_lisp("PhotonWidgets");

PtInit(nil);
win = new(PtWindow);
win.SetArea(100,100,100,100);

but1 = new(PtButton);
but1.text_string = "good button"
PtAttachCallback(but1,Pt_CB_ACTIVATE,#princ("good button\n"));
but1.SetPos(10,10);

but2 = new(PtButton);
but2.text_string = "error button"
PtAttachCallback(but2,Pt_CB_ACTIVATE,#g);
but2.SetPos(10,40);

PtRealizeWidget(win);

function interactive_mode ( level )
{
  local expr;

  princ("internal error: ", _last_error_,"\n")
  writec(stdout,\n  while ( (expr = read(stdin)) != _eof_)
  { try
    { writec(stdout,eval(expr));
      writec(stdout,\n    } catch
    { interactive_mode(level + 1);
      writec(stdout,\n    }
  }
}
```
while (t) {
    try {
        next_event();
    } catch {
        princ("starting temporary interactive mode using Lisp grammar\n");
        princ("use ^D to exit this mode and return to event loop\n");
        interactive_mode(1);
        princ("\nleaving temporary interactive mode\n");
    }
}
Functions and Program Structure

Function Definition

A function is defined in Gamma using a function statement:

```
function thing (a) { random() * a;}
(defun thing (a) (* (random) a))
Gamma> thing (5);
2.3869852581992745399
```

```
function pow (v, exp)
{
    local result = 1;
    while (exp ## > 0)
    {
        result *= v;
    }
    result;
}
```

No type specification is used since the type returned will be determined by the expressions within the function when it executes:

```
Gamma> pow (2,3);
8
Gamma> pow (2.1, 3.25);
19.4481
```

C programmers should note that this typeless function definition bears no similarity to a void function type, which does not exist in Gamma.

The value and type of a function is the return value of the last expression evaluated within the function. To return the value of a specific variable that only exists within the scope of the user function, that variable name is placed by itself on the last line of the function. This effectively causes that symbol to be evaluated, returning its value.

Functions do not need to be defined before they are referenced in a file, but they must exist before they are called. In other words, it is the run-time order, not the loading (reading) order that is important.
Function Arguments

When a function is called, the arguments in the call are mapped to the arguments specified in the function definition on a one-to-one basis. The Gamma engine evaluates the arguments and maps the results of those evaluations to each function argument name. Since Gamma is abstractly typed, there is no need to specify a data type in the function definition. If a particular data type is required within the function, then the function body can check for the type using the type predicate, \texttt{type-p}.

Variable number of arguments

It is possible to create a function that takes a variable number of arguments. The last argument in a function's argument list may be made to act as a "catch-all" or \texttt{vararg} argument which collects all remaining arguments provided in the function call as a list. For example,

\begin{verbatim}
function f (x, y...)
\end{verbatim}

creates a function with 2 mandatory arguments, the second of which can have one or more values. If this function is called as \texttt{f (1, 2)}, then \texttt{x} will have the value \texttt{1}, and \texttt{y} will have the value \texttt{(2)}, that is, a list containing one element whose value is \texttt{2}. If this function is called as \texttt{f (1, 2, 3, 4, 5)}, then \texttt{x} will be \texttt{1}, and \texttt{y} will be \texttt{(2 3 4 5)}, a list of four elements. If this function is called as \texttt{f(1)}, then an error would occur because \texttt{y} is not optional.

Optional arguments

Gamma allows optional arguments at the end of an argument list. An optional argument is specified by appending a question mark (\texttt{?}) to an argument in the function's argument list. All arguments after the first optional argument are implicitly optional as well. If the caller wants to provide a value to an optional argument, then the caller must also provide values for all preceding optional arguments. If an optional argument is not provided during the call, then the argument will take on the value \texttt{UNDEFINED}, which must be dealt with within the body of the function. A default value for an optional argument can also be provided in the function definition. For example,

\begin{verbatim}
function f (x, y?, z=5)
\end{verbatim}

creates a function with 1 mandatory argument and two optional arguments. The argument \texttt{y} has no default value, and \texttt{z} has a default value of \texttt{5}. This function could be called as \texttt{f(1)}, \texttt{f(1, 2)} or \texttt{f(1, 2, 3)}.

Protection from evaluation

Any function argument can be protected from evaluation by an exclamation mark (\texttt{!!}) before the argument name in the function's argument list. For example,
creates a function with two mandatory arguments, the second of which will not be evaluated when it is called. If this function were called as f (2+2, 3+3) then x would have the value of 4, and y would have as its value the expression 3+3. y could be evaluated using eval(y) to produce the value 6.

**Variable, optional, unevaluated arguments**

A variable argument can also be made optional. If so, and if it is not evaluated, then all the arguments which are collected into its list will not be evaluated either. For example,

```cogent
function f (!y...? = 17)
```

creates a function with one optional argument named y. The argument y is not evaluated, and may take any number of values, passed as a list. If no argument is specified to the function, then y will have the value of 17. If, instead of 17 the default is set to nil, no default will be assigned. This syntax effectively gives a way to pass a list of arguments of any length to a function.

**Examples**

The following program shows example functions with argument lists similar to those described above.

```cogent
#!/usr/cogent/bin/gamma

function variable_args (x, y...)
{
    princ("---- Output from variable_args(x, y...) ---- \n");
    princ("The first arg: ", x, "\n");
    with a in y do
    {
        princ("One of the variable args: ", a, "\n");
    }
    princ("\n");
}

function optional_args (x, y?, z=5)
{
    princ("---- Output from optional_args(x, y?, z=5) ---- \n");
    if (undefined_p(y))
    
        y = "This value has not been defined."

    princ("The first arg: ", x, "\n");
    princ("The second arg: ", y, "\n");
    princ("The third arg: ", z, "\n");
    princ("\n");
}
```
function no_eval(x, !y)
{
    princ("---- Output from no_eval(x, !y) ---- \n");
    princ("This argument was evaluated: ", x, "\n");
    princ("This argument was not evaluated: ", y, "\n");
    princ("\n");
}

function many_args (fixed_arg, !y?... = nil)
{
    princ("-- Output from many_args(fixed_arg, !y?... = nil) ---- \n");
    princ("fixed_arg: ", fixed_arg, "\n");
    princ("y: ", y, "\n");
    princ("The first y arg: ", car(y), "\n");
    with a in cdr(y) do
    {
        princ("The next y arg: ", a, "\n");
    }
    princ("\n");
}

variable_args("hello", 9, "world", 4 + 7, #x);
optimal_args(1);
optimal_args(1, 2);
optimal_args(1, 2, 3);
noc_eval(2+2, 3+3);
many_args("Fixed", "hello", 9, "world", 4 + 7, #x);

The output of this program is as follows:

    ---- Output from variable_args(x, y...) ----
The first arg: hello
One of the variable args: 9
One of the variable args: world
One of the variable args: 11
One of the variable args: x

---- Output from optional_args(x, y?, z=5) ----
The first arg: 1
The second arg: This value has not been defined.
The third arg: 5

---- Output from optional_args(x, y?, z=5) ----
The first arg: 1
The second arg: 2
The third arg: 5

---- Output from optional_args(x, y?, z=5) ----
The first arg: 1
The second arg: 2
The third arg: 3

---- Output from no_eval(x, !y) ----
This argument was evaluated: 4
This argument was not evaluated: (+ 3 3)

---- Output from many_args(!y?... = nil) ----
One of the args: hello
One of the args: 9
One of the args: world
One of the args: (+ 4 7)
One of the args: 'x

Function Renaming

When a function is defined, Gamma automatically assigns the function definition to the symbol that was provided as the function name, in the global scope. This does not mean that the symbol and the function definition are permanently related.

(A function definition is an independent data object which can be passed as an argument to a function or assigned to a symbol in any scope. For a C programmer this makes a Gamma function definition operate in much the same way as a function pointer in C. However, Gamma function definitions are much more versatile.)

It is possible to re-map a function definition at run-time to modify its behavior. For example, we may like to modify the function pow defined in the Function Definition section. We would like the improved pow to check the argument type and accept a string as its argument as well as a number. In Gamma, there are functions like int_p, real_p, and string_p that are used to determine the data type of a variable. (For the complete list of -p functions see Data Types and Predicates in the Reference Manual).

Thus, we rename the pow function defined in the section Function Definition to __pow and write the new version as follows:

```gamma
...__pow = pow;
function pow (v, exp)
{
    local result;
    if (string_p (v))
    {
```
result = "";
while (exp ## > 0)
    result = string(result, v);
}
else
    result = __pow(v, exp);
result;
}

Then the function call pow("hello", 3) will produce:

"hellohellohello"

Loading files

Files are loaded from the disk to memory using the load and require functions. The load function loads a Gamma file every time it is called. The require function checks to see if a Gamma file has been loaded, and if not, it loads it. The load_lisp and require_lisp functions do the same thing for files written in Lisp grammar. All of these functions take the name of the file, as a string, for their argument.

As a file is loaded by the Gamma engine, the require mechanism is used to access additional files. This is similar to the #include directive used in C programs, and likewise permits modularization of the application code. Note however that since Gamma is a runtime language, there is no equivalent to object modules of compiled languages. The require function therefore provides the sole mechanism for bringing together modules that define an application.

The pre-defined global variable _require_path_ contains a list of the paths to be searched to find the specified filename. This variable usually references the current directory, and the location for libraries. The list of paths can be augmented with:

_require_path_ = cons ("my_directory_name", _require_path_);

The pre-defined global variable _load_extensions_ contains a list of default extensions that are used by the require functions. Filenames with these extensions do not have to specify the full filename in the require argument. The variable is initialized to (".slg" ".lsp" "."), and can be augmented in the same way as _require_path_.

The main Function

In Gamma there is no requirement for a function named main as there is in C. As a program file is loaded, a call to a function at the outermost scope will in fact cause that function to be run at that point. In the same way, variable definitions and assignments at the outermost scope level are executed, effectively becoming globals. In most cases, the appl-
cation is initiated by calling the user's "top level mainline" function at the end of the file, or by entering a loop, such as an infinite event loop.

If a function is defined with the name `main`, then Gamma will automatically start to execute that function after the load is complete. This is equivalent to placing `main` as the last statement in the file. Note that `main`'s function definition must contain the keyword `function`, just like any other function definition. Since `argv` is available as a global variable, `main` does not require any arguments.

**Executable Programs**

Since the Gamma language is based on Lisp (ie. SCADALisp), programs can be written and executed using either Gamma or Lisp grammar. How to execute a Gamma program is discussed in Stand Alone Executable Programs in the Getting Started chapter of this Guide, as well as in the next two sections of this chapter.

Writing Lisp programs is beyond the scope of normal Gamma programming, but it may be useful from time to time to invoke a Lisp executable. This is done in a similar way to Gamma. Stand-alone programs will invoke the Lisp engine by using the following shell directive as the first line of the file:

```
#!/usr/cogent/bin/lisp
```

The Photon dialect is available through:

```
#!/usr/cogent/bin/phlisp
```

**Running a Gamma Program**

Gamma programs can be run in two ways: as a stand-alone executable, or by invoking Gamma from the command line.

1. For stand-alone executable program, the user simply types the name of the executable, possibly with command line arguments. The program invokes the Gamma engine through the shell `!#` directive (as explained in Executable Programs in the Getting Started chapter of this Guide).

2. In the case where there is no Gamma engine "embedded" into the program, the command `gamma` is available to run the executable. This command has several options, a few of which we mention here, and the rest of which are given in the `gamma` entry in the Reference.

   - `-h` gives a help message displaying all the options for this command.
   - `-c` declares all Gamma constants at startup. These constants can be viewed using the `apropos function`.
   - `-d` saves debugging information: the file name and line number.
\texttt{--F} declares all Gamma functions at startup. As with the \texttt{--C} option, these functions can be viewed using the \texttt{apropos} function.

The next section discusses command line arguments for a program, and how to access them within the program.

**Command Line Arguments**

Gamma provides a mechanism for accessing command line arguments. The symbol \texttt{argv} contains a list of the parsed command line arguments. Thus, if you have an application named \texttt{my_app} which takes two arguments \texttt{arg1} and \texttt{arg2}, then the executable invoked with:

\begin{verbatim}
my_app arg1 arg2
\end{verbatim}

will receive the following \texttt{argv}:

\begin{verbatim}
(my_app arg1 arg2)
\end{verbatim}

Like any list, the length of \texttt{argv} is simply \texttt{length(argv)}; The command line arguments can be accessed like any list, using any of the following sample approaches:

\begin{verbatim}
for ( i=0; i < length(argv); i++)
{
    arg = car (nth_cdr (argv, i));
    ... process arg ...
}
\end{verbatim}

or similarly, but more efficient:

\begin{verbatim}
while (length (argv) > 0)
{
    arg = car (argv);
    ... process arg ...
    argv = cdr (argv);
}
\end{verbatim}

which can also be expressed, still more efficiently, and without modifying the original \texttt{argv}, with:

\begin{verbatim}
for (i=argv; i; i = cdr(i))
{
    arg = car (i);
    ... process arg ...
\end{verbatim}
Object Oriented Programming

Classes are a powerful feature that helps users to organize a program as a collection of objects. Users that are familiar with C++ will find some syntax similar.

Classes and Instances

A class is a collection of variables and functions that, together, embody the definition of data type that is distinct and significant for the user's problem. Class functions are called methods. Class variables can be of the two kinds: attributes and class variables. Attributes are more common and do not require any special identifiers. Class variables are defined with the identifier static. We'll discuss class variables later in this chapter. Every class has a name. Here we define a class named Polygon, and give it four attributes:

```lisp
class Polygon
{
  sides;
  angles;
  dimensions = 2;
  color = nil;
}
```

Here is another example, Catalog, with one attribute, defined in interactive mode:

```lisp
Gamma> class Catalog { data;}
( defclass Catalog nil [ ] [ data ])
```

When you define a class in interactive mode Gamma returns an internal representation of its class definition in Lisp syntax. This definition is a list with the following elements: the defclass function, the class name, the parent class name (nil in this case), the class methods and class variables in one array (none in this example), and the class attributes in a second array.

Default values can be assigned to the attributes. For example, the Polygon class (above) has 2 dimensions and no color by default. The Catalog class has no default data values.

Instances

A class is an abstract data type. A class is used by constructing new instances of it. This is done using the function new:

```lisp
Gamma> pentagon = new(Polygon);
{ Polygon ( angles ) ( color ) ( dimensions . 2 ) ( sides )}
Gamma> autoparts = new(Catalog);
{ Catalog ( data )}
```
In this example, the class is Polygon and the newly-created instance of the class is pentagon. Or, the class is Catalog and the instance is autoparts.

The variables of the instances of a class are called instance variables. They correspond to the attributes in the class definition. In the Polygon class, for example, the instance variables are: sides, angles, dimensions, and color. Note that the function `new` returns the written representation of an instance, which consists of the class name and a list of instance variables. An instance variable with a default value is represented as a dotted list, such as (dimensions . 2) in our example.

In Gamma, to set or query the instance variable of an instance, the dot notation is used. Thus, each of the instance variables associated with the pentagon instance can now be set as follows:

```
Gamma> pentagon.angles = 108;
108
Gamma> pentagon.sides = 5;
5
Gamma> pentagon.color = "blue";
"blue"
Gamma> pentagon;
{Polygon (angles . 108) (color . "blue") (dimensions . 2) (sides . 5)}
```

Notice that internally Gamma holds a class instance and its instance variables together in curly braces. This is called literal instance syntax.

**Methods**

Methods are functions that are directly associated with a class.

We will create a Lookup method for the Catalog class. This method lets you look up an entry in the catalog by a key associated with the entry. In this example we implement our data as an association list, that is, a list whose elements are also lists, each of which contains exactly two elements: key and value. The library function `assoc_equal` returns the remainder of the association list starting at the element whose key coincides with the key in the argument of the method `Lookup`. Thus, `Lookup` returns the list associated with the key. The special keyword `self` is used when the instance refers to itself within the function:

```
method Catalog.Lookup (key)
{
    car(assoc_equal(key, self.data));
}
```
Note that the keyword `self` can be omitted and the call would look as follows:

```
  car(assoc_equal(key, .data));
```

The calls to class methods are made by instances, using the dot notation. For example, the instance `autoparts` created above can call the `Lookup` method as follows:

```
Gamma> autoparts.Lookup("muffler");
nil
```

Since the data attribute did not have a default value, the first time call to `Lookup` returns `nil`. In order to put data in the data list, we must create another method:

```lisp
method Catalog.Add (key, value)
{
  local i;

  if (i = .Lookup (key))
  {
    prin("The entry ", key, " already exists\n");
    nil;
  }
  else
  {
    .data = cons(list(key, value), .data);
  }
}
```

Notice that the `Add` method is using `Lookup` to determine whether or not the entry already exists in the association list. If so, it returns `nil`. Otherwise the new entry is added to the data list using the library function `cons`. The return value of a method is the return value of the last function executed within the body of the method.

Now we can add some data. For example, we can add an entry with the keyword "muffler" and the value 1, which is, for example, the number of mufflers in the stock:

```
Gamma> autoparts.Add("muffler", 1);
("muffler" 1)
Gamma> autoparts.Add("starter", 5);
("starter" 5) ("muffler" 1)
```

Now we can look up the entry for a muffler by the keyword:

```
Gamma> autoparts.Lookup("muffler");
("muffler" 1)
```
Note that the autoparts instance variables can be queried using the dot notation as follows:

```
Gamma> autoparts.data;
(("starter" 5) ("muffler" 1))
```

### Inheritance

Let us consider the following example where a new class is created:

```lisp
defclass Book Catalog [] [data (size . 0)]
```

The `Book` class is called a **derived** class and the `Catalog` class is called a **base**, or **parent** class for the `Book` class. In addition to having its own attributes, methods, and class variables, a derived class **inherits** all these things from the base class as well. For example, the `Book` class inherits the `data` attribute from the `Catalog` class:

```
Gamma> math = new(Book);
(Book (data) (size . 0))
Gamma> math.Lookup("Calculus");
nil
```

In this case it returns `nil` because no entry "Calculus" was added to the list of data. Now we can create an `Add` method for the `Book` class. This method adds an author and a publisher to the association list of data. If the `Add` operation is successful, the size of the list is incremented by 1. This `Add` method internally calls the `Add` method of the base `Catalog` class using the `call` function. We say that the derived class inherits implementation from the base class. If we were to change the way the `Add` method is implemented in the base class, the implementation would propagate to the derived class.

```lisp
method Book.Add (title, author, publisher)
{
    local pair = list(author, publisher);
} 
```
local result = call(self, Catalog, Add, title, pair);

if (result)
{
    .size+= 1;
}

The method `Add` can be evaluated as follows:

Gamma> math.Add ("Calculus", "Thompson", "Wiley");
1

It returns the size of the math catalog as the result of the last evaluated expression within the method. Now if we would like to search for the entry "Calculus", the method `Lookup` is evaluated as follows:

Gamma> math.Lookup ("Calculus");
(Calculus (Thompson Wiley))

Classes can be related by "is-a” relations, since one class is a derived class of the other. There can also be "has-a” relations between classes. Let us consider the following example:

class Figure
{
    color;
    height;
    width;
}

class Book_1
{
    size;
    figure = new(Figure);
}

Class Book_1 "has" an instance of class Figure as an attribute. In other words, class Book_1 contains one instance of the class Figure. Let us consider the connections between the methods of the two classes with the "has-a” relations. Suppose the following methods are defined:

method Figure.Show()
The method `Show` of the `Book_1` class internally calls the method `Show` of the `Figure` ("contained") class. We say that the `Book_1` class delegates its method to the `Figure` class. Thus, the effect of the delegation is implementation inheritance. It's true that to inherit implementation, the `Figure` class could be simply derived from the `Book_1` class and the "is-a" relations would be in effect. But then it would be impossible for an instance of the `Book_1` class to have several instances of the `Figure` class.

Note that in the definition of the `Book_1` class, the `Figure` class is instantiated, which makes the attribute `figure` an instance of the class `Figure`. Thus, if an instance of the `Book_1` class is created it can evaluate its `Show` method right away:

```java
...
mystery = new(Book_1);
mystery.Show();
...
```

However, if an instance of `figure` is not actually created in a `Book` class definition,

```java
class Book_2
{
    size;
    figure;
}
method Book_2.Show()
{
    .figure.Show();
}
```

then it has to be instantiated for each new instance of `Book_2` before any "delegation" will occur:

```java
...
mystery = new(Book_2);
mystery.figure = new(Figure);
mystery.Show();
...
Instance Variables

We recall that instance variables (ivars), are non-method items that make up an instance of a class. In the example below, the ivars of the instance `math` are `data` and `size`. To set or query the value of an ivar use the class instance and ivar in dot notation:

```
Gamma> math.size;
1
```

Gamma has the ability to add ivars to a class at any time, using the function `class_add_ivar`. As an example consider the `Catalog` class used in the above examples. Suppose we would like to have a variable which holds the date when a catalog is started:

```
Gamma> class_add_ivar(Catalog,#start_date);
nil
Gamma> Catalog;
(defclass Catalog nil [...][data start_date])
```

Once an instance variable has been added to a class, all new instances of that class created after the variable was added will receive the new ivar.

```
Gamma> math;
{Book (data) (size . 0)}
Gamma> cooking = new(Book);
{Book (data) (size . 0) (start_date)}
```

Class Variables

Class variables (cvars), are non-method items that permanently belong to the class in which they are defined. One can think of a class variable as named data associated with the class. There is only ever one copy of the variable. All instances of that class share that copy. All derived classes and all the instances of the derived classes share that one copy. It is like a global variable.

Gamma has the ability to add cvars to a class at any time, using the function `class_add_cvar`. Once a class variable has been added to a class it becomes available to all new instances of that class and the derived classes. However they do not get a private copy of that variable but share one and the same variable that belongs to the class. As an example consider the `Catalog` class and its derived class, `Book`, once more.

```
Gamma> class_add_cvar(Catalog,#capacity, 200);
200
```
We can see that the derived class `Book` does not have a private copy of the class variable `capacity`. However this variable is *available* for the derived class as well as for the instances of that class:

```
Gamma> Book.capacity;
200
Gamma> history.capacity;
200
```

To set or query the value of a cvar use the class name (or the instance name) and the cvar in dot notation. Remember, though, a change to the cvar in any class or instance of `Catalog` will change it for all classes and instances of `Catalog`:

```
Gamma> Book.capacity = 300;
300
Gamma> history.capacity;
300
Gamma> history.capacity = 400;
400
Gamma> Book.capacity;
400
Gamma> Catalog.capacity;
400
Gamma>
```

## Constructors and Destructors

A *constructor* is a method that is automatically run when a new instance of a class is made. A *destructor* is a method that is automatically run when the instance is destroyed. Constructors are called for all parent (base) classes of an instance starting with the root of the instance's class hierarchy. Destructors are called for all parent (base) classes of an instance starting with the class of the instance and proceeding toward the root of the instance's class hierarchy.

In Gamma these two methods take the special names `constructor` and `destructor`.

```
method Book.constructor ()
{
}
```
We'll now set the example variable `total_books` to 2 (since two have already been created: math and history):

```
Gamma> total_books = 2;
2
```

A new `Catalog` object can now be created, and the effect of the constructor and destructor observed:

```
Gamma> biology = new(Book);
{Book (data) (size . 0) (start_date)}
Gamma> total_books;
3
Gamma> biology = nil;
nil;
Gamma> total_books;
3
Gamma> gc();
166
Gamma> total_books;
2
```

The constructor worked as expected, but the destructor appears to have failed. Only after the `gc` function was called did the destructor get called. The `gc` function forces the garbage collector to run. When `biology` was set to `nil` the memory containing the previous definition for `biology` was left unlinked. Once this unlinked memory was recovered by the garbage collector, the destructor was called.

The frequency of the garbage collector running will depend on the program written in Gamma. The garbage collector can be forced to run by using the `gc` function. Occasionally, system activity may prevent it from running immediately, but the requirement to run is noted and it will do so at the next opportunity.

Classes are often used to keep track of real-world objects, and as such, it is important to keep statistics on these objects. One of the most common methods of doing this is by using constructors and destructors to increment and decrement a counter of the number of objects created or currently available.
Polymorphism

The concept of **polymorphism** has its roots in programming language design and implementation. A language is called polymorphic if functions, procedures and operands are allowed to be of more than one type. In comparison with polymorphic languages, there are languages called monomorphic, such as FORTRAN and C. Being monomorphic means that it is not possible, for example, to define two subroutines in FORTRAN with the same name but different number of parameters.

Overloading is a specific kind of polymorphism which Gamma supports.

**Operator Overloading**

*Operator overloading* allows the programmer to define new actions for operators (+,-,*,/, etc.) normally associated only with numbers. For example, the plus operator (+) normally adds only numeric variables. Operator overloading allows the user to define an alternate action to adopt when non-numeric variables are used in conjunction with an operator. The plus operator is often overridden so that strings may be concatenated using the syntax:

```plaintext
result = "hello" + " " + "there";
```

When overloading an operator in Gamma the developer must exercise extreme caution since operator overloading is achieved by redefining the operator itself. The typeless quality of variables in Gamma does not allow the interpreter to select an appropriate operator based on the types of the arguments.

Consider the following program fragment:

```plaintext
// Assign plus to a function called 'real_plus'.
real_plus = \+;

// Re_define plus to check for strings, and call
// string() or real_plus() depending on arg types.
function \+ (arg1, arg2)
{
    if (string_p(arg1) || string_p(arg2))
        string(arg1,arg2);
    else
        real_plus(arg1,arg2);
}
```

The first step in this example is to re-assign the functionality of the + operator to a function called real_plus. Notice that the backslash character is used to pass the + character explicitly. Without the backslash Gamma would interpret the plus character in the function definition statement as a mistake in syntax.
Once this assignment and definition are entered into Gamma the plus operator can be used with strings as well as with numbers:

\[
\begin{align*}
\text{Gamma}> & \ 5 + 4; \\
& 9 \\
\text{Gamma}> & \ "hello" + " \" + "there"; \\
& "hello there"
\end{align*}
\]

While it is convenient to set up overloaded functions in Gamma, remember that user functions are generally slower than Gamma's built-in functions.

**Binary Classes and User Classes**

Binary classes are classes that are built into the specific version of Gamma that you are using. User classes are those classes defined by the programmer in the process of developing an application.

To test the number of built-in classes in the version of Gamma you are using, start a fresh instance and use the \texttt{apropos} function interactively to find all available classes:

```
andrew:/home/andrew > gamma
Gamma;(TM) Advanced Programming Language
Copyright (C) Cogent Real-Time Systems Inc., 1996. All rights reserved.
Version 2.4 Build 139 at Jul 6 1999 10:48:51
Gamma> apropos("*",class_p);
(Osinfo)
```

As we can see, \texttt{gamma} does not have built-in binary classes. Now let us try to run \texttt{phgamma}:

```
andrew:/home/andrew > phgamma
Gamma(TM) Advanced Programming Language
Copyright (C) Cogent Real-Time Systems Inc., 1996. All rights reserved.
Version 2.4 Build 139 at July 6 1999 14:21:45
Gamma> apropos("*",class_p);
(Osinfo PhArea PhBlitEvent PhBoundaryEvent PhDim PhDragEvent PhDrawEvent PhEvent PhEventRegion PhExtent PhImage PhKeyEvent PhLine PhPoint PhPointerEvent PhPrect PhRect PhRegion PhRgb PhWindowEvent PtArc PtBarGraph PtBasic PtBasicCallback PtBezier PtBitmap PtBgkd PtButton PtCallbackInfo PtComboBox PtComboBoxListCallback PtComboBoxTextCallback PtContainer PtDivider PtEllipse PtEventData PtFontSel PtGauge PtGenList PtGenTree PtGraphic PtGrid PtGroup PtHtml PtIcon PtLabel PtLine PtList PtListCallback PtMenu PtMenuBar PtMenuButton PtMenuLabel
```
There is a significant difference in supported classes between the `gamma` and `phgamma` executables. The reason is that Photon widgets are mapped into `phgamma` as classes. The standard Gamma executable does not have support for Photon graphics and does not have these built-in binary classes.

User classes are found in the same manner. After user classes are defined they will match the `class_p` predicate in the `apropos` function and be added to the list:

```lisp
(class test
  { a; b; c; }
(defclass test nil [] [a b c])
```

```
Gamma> class test
Gamma> (defclass test nil []) [a b c])
```

```lisp
Gamma> apropos("*",class_p);
(Osinfo test)
```
Tutorial III

Classes and OOP

Gamma implements object-oriented programming (OOP) features which provide a single-inheritance class mechanism with instance variables and methods. Since Gamma is an interpreter, the object definitions are truly dynamic, allowing for run-time extensibility. This example provides the simplest of starting points to this key software methodology.

```gamma
#!/usr/cogent/bin/gamma

/*
 * Demonstrates:
 * class definitions: attributes and methods.
 * constructors and destructors
 * method overloading
 */

/*
 * Define a class of animal, with no default type and a default
 * of 4 legs
 */
class animal
{
    type = "animal";
    num_legs = 4;       // By default, animals have 4 legs
}

/*
 * The constructor for an animal is called when any instance of
 * animal or a subclass of animal is created using a call to 'new'.
 * Constructors have no arguments.
 */
method animal.constructor()
{
    princ ("A ", class_name(class_of(self)), " is born
");
}

/*
 * The destructor for an animal is called when any instance of animal
 * or a subclass of animal is deleted by the garbage collector.
 * There is no explicit deletion mechanism in Gamma. Destructors
 * have no arguments.
 */
method animal.destructor ()
```


`{  princ ("A ", class_name(class_of(self)), " dies\n");}

/*
 * All methods except constructor and destructor are overloaded,
 * meaning that only the method for the nearest class in the
 * ancestry of the instance will be called for any given method name.
 */
method animal.describe ()
{  princ ("The ", self.type, " has ", self.num_legs, " legs.\n");}

/*
 * Create a subclass of animal of a particular type.
 */
class cat animal
{
  type = "feline";
}

/*
 * Create another subclass of animal which is itself a parent class
 */
class insect animal
{
  num_wings = 2;
  num_legs = 6;
}

/*
 * Overload the description method for insects so we hear about
 * wings and legs when we ask about insects.
 */
method insect.describe ()
{
  /*
   * We can explicitly call a method of a parent class using the
   * 'call' function and naming a parent class.
   */
  call (self, #animal, #describe);
  princ (" (oh, and ", self.num_wings, " wings)\n");```
method insect.destructor ()
{
    princ("Crunch. ");
}

class beetle insect
{
    type = "rhinoceros beetle";
    num_wings = 4;
}

function main ()
{
    local  pet = new (cat);
    local  bug = new (beetle);

    /*
    * cat gets its describe method from the animal class
    */
    pet.describe();

    /*
    * beetle gets its describe method from the insect class
    */
    bug.describe();

    /*
    * Since the destructor will be implicitly called by the garbage
    * collector, we can cause the destructor to occur by removing
    * all references to the instances (set the variables referencing
    * the instances to nil), and then explicitly invoke the garbage
    * collector. Typically this is not necessary, as the garbage
    * collector will run when necessary.
    */
    pet = nil;
bug = nil;

gc();
}
Interactive Development and Debugging

Interactive Mode Implementation

The implementation of Gamma's interactive mode provides an interesting example of the
how to use the concise power of the language. Interactive mode is implemented with the
following few lines of Gamma:

```gamma
c princ("Gamma> ");
 flush(stdout);
 while ((x = read( stdin)) != _eof_)
  {
    princ( eval( x));
    terpri();
    princ("Gamma> ");
    flush(stdout);
  }
```

An application can easily provide its own customized "interactive mode" by executing a
Gamma script file with a variation of this code that is entered when the file is loaded.

Getting On-Line Help for Functions

Gamma can display function definitions and parameters. To do this, start Gamma interac-
tively and type the name of the function followed by a semicolon and return. For example,

```
anrewt@1:~ > Gamma
Gamma (TM) Advanced Programming Language
Copyright (C) Cogent Real-Time Systems Inc., 1996. All rights
reserved.
Gamma> init_ipc;
(defun init_ipc (my_name &optional my_queue_name domain) ...)
Gamma> new;
(defun new (class) ...)
Gamma> array;
(defun array (&optional &rest contents) ...)
Gamma> insert;
(defun insert (array position_or_function value) ...)
```

Note that the function definitions are described in the internal Lisp representation, as a
list. The function is always displayed with the word defun first, followed by the name of
the function, and then its syntax. The function arguments are enclosed in parentheses,
but not separated by commas as they are in Gamma syntax. The Gamma function mod-
ifiers (!, ?, and ...) are represented by: &noeval, &optional, and &rest respectively.
For details on these modifiers, see function in the Reference section. To give you a general idea, here is how the above functions definitions appear, first in Gamma syntax and then Lisp syntax:

```
init_ipc (my_name, my_queue_name?, domain?)
(defun init_ipc (my_name &optional my_queue_name domain) ...)

new (class)
(defun new (class) ...)

array (s_exp?...)
(defun array (&optional &rest contents) ...)

insert (array, position|compare_function, value)
(defun insert (array position_or_function value) ...)
```

### Examining Variables in a Class or Instance

Classes and instances can be examined in two ways. For a class, you can simply type the name at the prompt. Instances of classes bound to C structures can be viewed using the `instance_vars` function. To examine an instance of a class, simply type an expression which evaluates to that instance (see, for example, `elephant` as an instance of the `animal` class in the Instances section of the Class chapter).

You can examine not only user-defined classes, but also the classes which are implemented in Gamma. For example,

```
Gamma> PhImage;
#< Binary Class: PhImage >
Gamma> instance_vars (PhImage);
[bp1 colors flags format image image_tag palette palette_tag size \
  type xscale yscale]
Gamma> x = new (PhImage);
(PhImage (bp1 . 0) (colors . 0) (flags . 0) (format . 0) \
  (image . #{})) (image_tag . 0) (palette . []) (palette_tag . 0) \
  (size . (PhDim (h . 0) (w . 0))) (type . 0) (xscale . 0) \
  (yscale . 0))
Gamma> x;
(PhImage (bp1 . 0) (colors . 0) (flags . 0) (format . 0) \
  (image . #{})) (image_tag . 0) (palette . []) (palette_tag . 0) \
  (size . (PhDim (h . 0) (w . 0))) (type . 0) (xscale . 0) \
  (yscale . 0))
Gamma> instance_vars (x);
[(bp1 . 0) (colors . 0) (flags . 0) (format . 0) (image . #{})) \
  (image_tag . 0) (palette . []) (palette_tag . 0) (size . \
  (PhDim (h . 0) (w . 0))) (type . 0) (xscale . 0) (yscale . 0)]
Gamma> pretty_princ (x, "\n")
(PhImage (bp1 . 0) (colors . 0) (flags . 0) (format . 0) (image . #{}))
```
Using the Debug Prompt

The `debug>` prompt appears when an error occurs in interactive mode. Gamma halts execution of the program and produces the prompt. You can perform any action at the `debug>` prompt that you can perform at the top level, including modifying program source and setting variable values. The value of any variable can be queried by simply typing its name. The calling stack can be queried by using the `stack` function.

The `stack` function displays the execution stack, providing a list with the names of the nested functions executing when the error occurred. The outermost, or top level, function appears first (after `progn`). The function causing the error appears last on the list.

Once the `debug>` prompt appears, the program cannot be continued and must be restarted. If an error occurs again as a result of code executed within the debug level, another nested level of debug will appear. Each level adds to the current point on the execution stack. You can move up debug levels and return to the Gamma> prompt by pressing `Ctrl-D` at the `debug>` prompt.

Debugging a program

The use of an interpreter engine enables some unique approaches to the process of debugging and testing software. This section describes some of the tools and techniques for debugging an application.

Interacting with an Active Program

Gamma can provide an active `view-port` into the running application. Another task (or shell) can, at any time, interact with a running Gamma program, without halting it or otherwise disturbing its real-time response. This provides an approach to debugging that is much more powerful than adding debug print statements.

For example, suppose that we started a process with the name "my_task" interactively:

```
Gamma> init_ipc ("my_task");
```

The `gsend` (for Gamma) or `lsend` (for Lisp) utility is used to interact with a running application from a shell:

```
[sh]$ gsend my_task
my_task>
```
The `lsend` utility accepts Lisp input as the default and `gsend` accepts Gamma input as its default.

Once connected to a running Gamma program using `gsend/lsend`, the developer can:

- query/set variables/objects/instance_vars in the global scope
- call functions/methods
- re-define function definitions
- run any Gamma command interactively

The syntax for starting the `gsend` utility is as follows:

```plaintext
gsend [-l] [-g] [program] [pid]
```

- `-l` Accept Lisp input from the keyboard.
- `-g` Accept Gamma input from the keyboard.
- `program` a Gamma program name, attached by name_attach, init_ipc, or qnx_name_attach
- `pid` (QNX 4 only) a task ID

`gsend` and `lsend` attach to a running Gamma program and allow the user to send commands without exiting the event loop of the attached process. Any statement may be issued, including changing the definitions of existing functions. In our simple example we can call the `princ` function for `my_task` to execute:

```sh
$ gsend my_task
my_task> princ ("Hello!\n");
t
```

Notice that event processing stops for the duration of the command. Now let's look at `my_task`. In order to respond to requests from `gsend/lsend`, `my_task` must be executing an event loop. We can start one using the `next_event` function in a `while` statement:

```gamma
Gamma> init_ipc ("my_task");
t
Gamma> while(t) next_event();
Hello!
```

The `my_task` program continues to run as normal during this operation.
This presents an excellent opportunity for rapid development by programmers. Typically developers are used to the "code, compile, link, run, debug, code..." iterative approach to programming. Once a Gamma developer makes a program with a well written event loop, such as the one shown in the section below on trapping errors, programming and testing can become operations that happen in parallel.

Programmers will find that after a piece of code has been written, it can be uploaded to an already running Gamma process with `gsend/lsend` by using a simple "cut and paste". The code is automatically assimilated into Gamma and ready to run. Better yet, if there is a problem with the code, the programmer receives immediate feedback and can track the problem down through an interactive debugging prompt that can be built right into the event-loop.

### Trapping and Reporting Errors

Gamma provides a pair of functions referred to as the `try/catch` mechanism, that is very important for debugging. Consider the following simple event loop:

```c
while (t)
{
    next_event();
}
```

This will run until the program exits or an event triggers some code that produces an error condition. There is no protection against errors. Now consider the following setup:

```c
while (t)
{
    try
    {
        next_event();
    }
    catch
    {
        prin("error occurred\n");
    }
}
```

This setup of `try/catch` will try to evaluate the block of code contained in the "try" portion and jumps to the "catch" portion when an error occurs. A more effective example of the `catch` code block is:

```c
while (t)
{
    try
    {
```
This setup provides the developer with information about the last error and the calling stack which led to the last error. Tutorial II provides an example which illustrates the try/catch and protect/unwind mechanisms to get reports on an error.

Another setup that the developer may find useful is to automatically start an interactive session in the case of an error. The example of such a setting can be found in Tutorial II.

Determining Error Location

The stack function will show the current function calling stack, expressed as a list of functions that the interpreter is currently evaluating. To trace the execution path of parts of a program it is useful to print out the code as it is evaluated. The trace and notrace functions act as delimiters to areas when tracing should occur. The tracing information is delivered to standard output.

The following table of predefined global variables provides additional information useful for debugging:

Table 1. Global Variables in Gamma

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>error_stack</em></td>
<td>The stack at the time the last error occured.</td>
</tr>
<tr>
<td><em>unwind_stack</em></td>
<td>The stack at the time that an error was discovered.</td>
</tr>
<tr>
<td><em>last_error</em></td>
<td>A string containing the last error.</td>
</tr>
</tbody>
</table>

Filtering Object Query Output

Gamma permits the user to control the level of detail reported, and the format used, when an object is queried. This is done by defining a function named _ivar_filter with two arguments. For example, each class instance has a number of instance variables that are reported during interactive mode in the format:

Gamma> stats = qnx_osinfo(0);
(Osinfo (bootsrc . 72) (cpu . 586) (cpu_speed . 18883) (fpu . 587) (freememk . 16328) (machine . "PCI") (max_nodes . 7) (nodename . 2) (num_handlers . 64) (num_names . 100) (num_procs . 500) (num_sessions . 64) (num_timers . 125)
The following example provides a function named `ivar_filter` that controls the output format. Note that each instance variable consists of a name and a value. If we define the following:

```lisp
function _ivar_filter (!instance,!value)
{
    princ(format("%20s %20s", string(car(value)), string(cdr(value))));
    nil;
}
```

then the output for Gamma in interactive mode now looks like:

```
Gamma> stats;
(Osinfo
    bootsrc 72
    cpu 586
    cpu_speed 18883
    fpu 587
    freememk 15544
    machine PCI
    max_nodes 7
    nodename 2
    num_handlers 64
    num_names 100
    num_procs 500
    num_sessions 64
    num_timers 125
    pidmask 511
    release 71
    reserve64k 0
    sflags 28675
    tick_size 9999
    timesel 177
    totmemk 32384
    version 423
)
Advanced Types and Mechanisms

Symbols

We introduced symbols in the Symbols and Values section at the beginning of this Guide. Recall that a symbol is a unique word or name within the system, and that references to a particular symbol will be to the exact same Gamma object, regardless of how that reference was obtained. Symbols can be used as variables, as they may have a value that can be queried through evaluation. The value of a symbol can change depending on the current execution scope.

A symbol may contain any character, though it is necessary to escape some characters using a backslash (\) when writing them. The normal character set for symbols consists of the following:
- The lowercase letters (a-z)
- The uppercase letters (A-Z)
- The digits (0-9)
- The underscore (_)

A symbol is created by a call to symbol, or by reading any legal string of characters which forms a symbol.

Undefined symbols

In Gamma a variable must be assigned a value. A variable does not exist until a value is assigned to it. Once defined, both the value and type of a variable can be changed, effectively re-defining the variable. A variable which is used but has never been assigned a value will cause an error:

```
Gamma> 3 + k;
Symbol is undefined: k
debug 1>
```

The undefined_p function can be used to test if a variable is defined, as follows:

```
Gamma> undefined_p (a);
t
Gamma> a = 5;
5
Gamma> undefined_p (a);
il
```

Uniqueness of Symbols

The uniqueness of symbols in the system provides an interesting way to perform the equivalent of the C language enumerated type, in case you want a list of constant values
representing different things. In Gamma, when two symbols are tested for equality, the comparison is first done on the symbol reference itself, not the value associated with the symbol. This is because the Gamma \(==\) operator is mapped to the Lisp \texttt{equal} function, which determines equality first with the \texttt{eq} function. The Lisp \texttt{eq} function tests for equality of the reference, and only if this has failed will the \texttt{equal} function perform an equality test on the value of the references. Also in Gamma, a symbol can be defined without assigning a value.

When the Gamma engine reads a literal symbol (see \textit{Literal Syntax and Evaluation} in this chapter), as illustrated in the example below, it determines that the reference is in fact a symbol. If the symbol does not exist, Gamma creates the it with a value of \texttt{_undefined_}.

\begin{verbatim}
Gamma> x = #yes;
yes
Gamma> yes;
Symbol is undefined: yes
debug 1>
\end{verbatim}

Therefore, it is valid in Gamma to make comparisons for equality between symbols whose values are not defined. Such comparisons between symbols are actually more efficient than comparing the values of two symbols. This leads to the following example. Here Gamma uses the equivalent of an enumerated type, but it is more efficient than assigning actual values to the constants, since the test is for the symbol reference only.

\begin{verbatim}
Gamma> function my_state (x)
{
    if (x==#on)
        princ ("I am on.\n");
    else if (x==#off)
        princ ("I am off.\n");
    else if (x==#unstable)
        princ ("I am not stable.\n");
    else
        princ ("I don't know.\n");
}
(defun ........)
Gamma> a = #on;
on
Gamma> my_state (a);
I am on.
t
Gamma> my_state (#off);
I am off.
t
\end{verbatim}
Notice that the enumerated set (on, off, unstable) was not created as variables with assigned values (1, 2, 3,...) but used directly, leading to a more efficient and cleaner implementation.

Properties

Symbols can be assigned properties, using the `setprop` function. Each assigned property is a name/value pair, which is globally defined for the symbol, regardless of the current scope and value. These properties can be accessed using the `getprop` function.

Predefined Symbols

There are some symbols, such as `_undefined_` mentioned above, whose values are predefined in Gamma. For the complete listing of symbols that are predefined in Gamma see section Predefined Symbols in the Reference Manual.

Evaluation

Expressions in Gamma consist of a few basic types, which are submitted to an evaluator to produce a return value. Every evaluation returns a result. Any Gamma object may be submitted to the evaluator. The evaluator behaves differently according to the type of the evaluated object.

**Table 2. Type Evaluation**

<table>
<thead>
<tr>
<th>Type of Object</th>
<th>Evaluation result</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbol</td>
<td>The value of the symbol in the current scope.</td>
</tr>
<tr>
<td>list</td>
<td>Function or method call.</td>
</tr>
<tr>
<td>all others</td>
<td>Itself.</td>
</tr>
</tbody>
</table>

Evaluation of a Symbol

If a symbol is being used as a variable, then its value will depend on the scope in which it is being evaluated. A new scope is entered whenever a user-defined function is executed, or when a `with` statement is executed. A symbol is defined within a scope when it appears in the argument list of a function, or when it appears in the variable list of a `local` statement. If a symbol is not defined within the current scope, then the value of the symbol in the next most local scope is used. This is called dynamic scoping, since the scope of a symbol used in a function may depend on the calling sequence that executed that function. The value of a symbol may be any Gamma object. See examples on dynamic scoping in Tutorial II.

Evaluation of a List

Since Gamma is built on top of Lisp, all Gamma statements and expressions are translated into lists, which represent the equivalent function call. For example, the expression 2 +
3; is translated into the function call \((+ 2 3)\) at read time, and evaluated as a list at run time. (Operators like + are functions in Lisp.) For more details on Lisp function syntax, see Getting On_Line Help for Functions.

When a list is submitted to the evaluator, it will be treated as either a function call or a method call. The first element in the list is evaluated, and the result examined. If the first element evaluates to a function definition, then the remainder of the list is treated as the arguments to this function, and the function is called. If the first element evaluates to an instance of a class, then the second element is evaluated.

If the second element evaluates to a class, then the third argument must be a symbol that names a method for that class. If the second element does not evaluate to a class, then it must be a symbol that names a method for the class of the instance, or a method of one of the instance's parent classes.

Once a method has been identified, the remaining elements of the list are treated as the arguments to the method, and the method is called on the instance. For example, here we call a function with the given arguments:

\[(\text{function arg1 arg2...})\]

Here we call a method from an instance's own class.

\[(\text{instance method_name arg1 arg2...})\]

And here we call a method from the instance's class hierarchy.

\[(\text{instance class method_name arg1...})\]

If an explicit class is provided, it is normally a parent (base) class of the given instance. This is not enforced by the evaluator, so it is possible to call a method for an instance which is not a member of the class for which the method was defined. This is not normally a good idea, and can be highly confusing to anybody reading the code.

**Evaluation to Itself**

Most Gamma object types evaluate to themselves, meaning that the result of submitting the object to the evaluator is the object which was submitted, unmodified. Any object except a list or a symbol will simply return itself when evaluated. For example, the number 5, when evaluated, will return 5, which is itself.

**Literal Syntax and Evaluation**

One of the most powerful features of an interpreter-based language such as Gamma is the ability to evaluate symbols and expressions at run-time. Gamma uses the # operator
to indicate a literal expression. For those familiar with Lisp, this is equivalent to the forward quote syntax. Gamma also supports evaluation of sub-expressions, using the ` and @ operators. For more details on their use, see Quote Operators and further explanation below.

**Literal Expressions**

A literal expression is the expression that specifies an actual value rather than a function for creating the value. For example, the number 3 is a literal, where the expression (+ 1 2) is not. Similarly, the string "hello there" is a literal, and the expression string("hello ", "there") is not, yet they produce equal values when evaluated.

Most object types in Gamma have Lisp and Gamma literal forms. You can create a valid object of some types (such as numbers, symbols, and strings) by reading a literal from a file or the command line. Other types (such as arrays, classes, and functions) are created by corresponding statements or functions.

See Literals in the Reference Manual for definitions, notations, and examples of literal expressions in Gamma.

**Deferring Expression Evaluation**

A Gamma expression preceded by the quote operator (#) will be taken literally, i.e., it will be protected from the evaluator. When the symbol containing this literal is evaluated, its contents are then interpreted. For example:

```
Gamma> x = #5 + 6;
(+ 5 6)
Gamma> #x;
x
Gamma> x;
(+ 5 6)
Gamma> eval (x);
11
```

In the first case, the quote operator (#) protects the entire expression from the evaluator. That is, it protects everything to its right, all the way to the end of the expression (usually a semicolon or closed parenthesis). In the second case it is used to "produce" the literal symbol x. Then x is evaluated, returning its literal contents. Finally, the eval function is used to force execution of the literal contents of x. The eval function forces the resolution of variable references, as in this example:

```
Gamma> a = 1;
1;
Gamma> x = a + 5;
6;
Gamma> x = #a + 5;
```

The literal is often used to delay the evaluation of an expression until an event is triggered. A good example is the `add_set_function`. This function takes two arguments. The first argument must be a symbol, so the `#` operator is used to prevent the required symbol from being evaluated. The second argument is simply any expression, most commonly a function. The `add_set_function` function sets the second argument to be evaluated when the first argument is changed:

```
Gamma> add_set_function (#a, #princ("My value = " ));
(princ "My value =")
Gamma> a = 21 / 3;
My value = 7
```

In the following variation of the above example, a symbol used as an argument has been assigned a literal symbol, so that its evaluation will result in the desired symbol:

```
Gamma> x = #b;
b
Gamma> add_set_function (x, #princ("My value = " ));
(princ "My value =")
Gamma> b = 21 / 3;
My value = 7
```

### Literal Function Arguments

Some functions require arguments that are symbols. Normally, the arguments to a function are evaluated before the function is actually invoked. It is possible to cause a function’s arguments not to be evaluated by using the `exclamation` modifier in the function declaration.

As an example, we can write our own version of the `add_set_function` function mentioned above:

```
Gamma> function my_add_set (!sym, !exp)
    {add_set_function(sym, exp);}  
(defun my_add_set (&noeval sym &noeval exp) (add_set_function sym exp))
Gamma> my_add_set (c, princ("My value = "));  
(princ "My value = ")
Gamma> c = 21 / 3;
My value = 7
```
For more details on function arguments see Function Arguments in the Functions and Program chapter of this Guide.

**Partially Evaluated Literal**

Gamma supports evaluation of sub-expressions, allowing you to write expressions whose elements may or may not be evaluated. In the following example, we want to create a literal expression which will calculate \( a \) to the power of \( b \), where \( b \) is a specific power, evaluated at the time the literal is defined. The operator `\` is used like \# to prevent evaluation of the expression, but it allows for exceptions. These exceptions, which will be evaluated, are denoted using the @ operator.

```
Gamma> b = 3;
3
Gamma> my_cube = `(pow (a, @b));
(pow a 3)
Gamma> a = 10;
10
Gamma> eval(my_cube);
1000
```

In the following example, the timer event is used to demonstrate how the current value of a variable can be evaluated into a literal:

```
Gamma> a = "hello";
"hello"
Gamma> every (15, #princ(a,"\n"));
hello
nil
Gamma> a = "goodbye";
"goodbye"
Gamma> next_event();
goodbye
Gamma> cancel (1);
[866239787 836823463 15 ((princ a,"\n")) 1]
Gamma> every (15, list (#princ, a, "\n"));
goodbye
Gamma> next_event();
goodbye
nil
Gamma> a = "no more";
"no more"
Gamma> next_event();
```

Constructing Variable Names at Run-time

Controlling when an expression is evaluated lets you generate the actual variable names at run-time. This can produce extremely concise code, particularly compared to the C language equivalent. In the following example, a set of simple objects each has a value. The object name and its value is entered. In a conventional language, we might search the array of objects to find the one with the given name, and then make the assignment. Gamma makes it possible to directly construct the variable reference using the set function, as follows:

Gamma> name = "fido";
"fido"
Gamma> value = "bites";
"bites"
Gamma> set(symbol(string(name)), value);
"bites"
Gamma> fido;
"bites"

Note that the syntax does not accept the = assignment operator, so the functional form of the assignment operator: set must be used. Note also that we would probably use undefined_p to verify that the variable actually existed to avoid halting the program due to an undefined variable error. Although the example is trivial, this technique is very useful for constructing function references based on run-time data.

Literal Array Syntax

An array is defined in Gamma with the array function, which creates the array and sets the elements to the specified values.

Gamma> x = array (3, "hi");
[3 "hi"]
Gamma> x[3] = #a + 5;
(+ a 5)
Gamma> x;
[3 "hi" nil (+ a 5)]
Generally, literal arrays should be avoided except for static variables. A literal array is embedded into your code. If it is changed, then the code is effectively changed!
Input and Output

Referencing Files

As in C, there are two ways to reference a file in Gamma, using a descriptor or a pointer.

A file descriptor is an integer that identifies an open file within a process. This is the lowest-level handle available for interacting with the open file. Disk files, pseudo-ttys, IP sockets, UNIX-domain sockets, pipes and other facilities all offer interaction through file descriptors.

A file pointer is an abstraction of the file that adds buffering on input and output. This would be of type FILE* in C. The reason this exists is that it is very inefficient to use a file descriptor, which does not perform any in-process buffering where many reads and writes are being performed. The file pointer stores many write requests until it has enough data to perform a more efficient write to disk, hopefully in multiples of the disk block size.

In Gamma, a file pointer is an opaque structure (the internals are not visible to the programmer) that is effectively a buffered file. (See the note in open.) It’s abstracted a little further to also include strings as file pointers, when they are opened using open_string.

Some Gamma I/O functions work with file descriptors (generally those that start with fd_), others work with file pointers, and a few work with both.

Lisp and Gamma I/O mechanisms

Gamma provides sophisticated mechanisms for reading and writing expressions, which can greatly simplify most file manipulation functions. There are two fundamental facilities for manipulating file data in Gamma: the reader and the writer. Gamma is based on the SCADALisp engine, and acts as a read-time translator from Gamma syntax to SCADALisp internal form. Thus expressions can be read in either Gamma or Lisp (we abbreviate SCADALisp as simply Lisp) representation.

Since the internal representation of an expression is an optimized Lisp mapping, the writer will produce its output as Lisp. This makes the reader and the writer symmetrical in Lisp, but not symmetrical in Gamma. A purely symmetrical Gamma writer is not possible, since there is no way to express literals in Gamma for data types such as list, buffer, array, instance and class.

The Lisp writer is aware of the format that the Lisp reader requires, and is able to format any expression such that the reader can subsequently read it back in. This means that an arbitrarily complex expression, such as a list containing instances of a class whose instance variables include arrays and other instances, can be written using a single line of code, and read back in using a single line of code as well. Since a Gamma function is simply a data object, it can be written and read in exactly the same way.
Generally, the lack of symmetry between the Gamma reader and the Lisp writer is not a problem, since any data written by Gamma will still be readable simply by instructing the open function to recognize Lisp instead of Gamma syntax.

Writing

Print vs. Princ

It is not always appropriate to write a data item in a way that can be read by the Lisp reader. For example, the Lisp reader requires that all character strings are surrounded by double quotes to differentiate them from symbols and to deal with white space and special characters. In some cases, the programmer may wish to write a character string in "human-readable" form, with no quotes and escapes on special characters.

The Gamma writer will produce both kinds of output. The print function will always generate output which can be read by the Lisp reader, including escape characters, quotation marks and buffer and instance special forms. The princ function attempts to make the output as readable as possible to a human, but will not necessarily produce output that can be read by the Gamma reader. The name princ is historical, and can simply be thought of as an alternate form of print. Notice that neither princ nor print will automatically place a carriage return at the end of a line. The programmer must explicitly print a "\n" or make a call to terpri.

Write vs. Writec

Like princ and print, there are two forms of the write function. The write function operates identically to the print function, except that its first argument declares the file handle to which it will write its output. The result of a write function is machine readable, whereas the result of a writec function is intended to be human readable. Notice that neither writec nor write will automatically place a carriage return at the end of a line. The programmer must explicitly print a "\n" or make a call to terpri.

Terpri

The terpri function will produce a carriage return either to the standard output or to a given file handle. terpri is most commonly used to generate a carriage return in a file that is being written using the write function. If the programmer were to use (write file "\n") then the file would actually contain the four characters "\n", rather than the intended carriage return. terpri will insert a carriage return into the file under any circumstances.

Pretty Printing

All of the printing functions have a further variant, known as the pretty printing functions. These variants attempt to format the output to an 80-column page, inserting line breaks and white space in order to make the output more readable. The pretty-printing indentation rules are intended to make data structure and program flow more easily understood, and closely follow the pattern used by GNU Emacs in its Lisp indentation mode.
Printing Circular References

It is common when programming with dynamic lists and arrays, or when constructing inter-related class definitions, to create a data structure which is self-referential, or which contains circular references. For example, it may be useful to have a child class contain a pointer to its parent, and the parent class contain a pointer to its child. In this circumstance, an attempt to print an instance of the child class would cause the Lisp writer to enter an infinite loop if it did not take precautions. In C programs, this circumstance is normally avoided by having a printing routine which understands the child/parent relationship and simply writes them in such a way that the infinite loop is never entered. This carries the problem that each data structure must have its own dedicated printing routine, which necessarily does not preserve a generalized data syntax, and which cannot perfectly represent the child/parent relationship in any but the simplest of cases.

Gamma solves the problems of self-reference and circularity by modifying the printed representation of an object to include embedded reference points in the data structure. Whenever a Gamma object is printed, all circular references and self-references are detected before the object is printed, and reference points are inserted into the printed representation. Subsequent attempts to print an object that was previously printed will merely produce a reference to the first printing of the object. This facility produces a result that is essentially impossible in languages such as C; it perfectly preserves multiple pointer references to data which are not known, a priori, to be multiple references.

A very simple example of self-reference may be a list that contains itself. This is normally achieved using destructive functions such as nappend, rplaca and rplacd. Consider the following dialogue:

Gamma> a = list(1, 2, 3);
(1 2 3)
Gamma> rplacd (cdr (a), a);
(1 2 (1 2 (1 2 (1 2 (1 2 (1 2 (1 2 ...)))))))))))))

In this case, by replacing the tail of the list with the list itself, it is possible to create a self-referential list which cannot be printed using normal means. Any attempt to print this list will cause an infinite loop. The Lisp writer in fact produces the following output:

Gamma> a = list(1, 2, 3);
(1 2 3)
Gamma> rplacd (cdr (a), a);
#0=(1 2 . #0#)

The first time that the self-referential list is printed, the Gamma writer determines that a self-reference will occur, and marks that point with a numbered place holder, using the syntax #n=, where n is a monotonically increasing number counting the number of circular references in the data object. Each subsequent reference to the marked object will cause the writer to produce a reference back to the original using the syntax #n#. For ex-
ample, if we create another, similar list, and then put both lists together into another list, we will get the following:

```
Gamma> b = list(4, 5, 6);
(4 5 6)
Gamma> rplacd (cdr (b), b);
#0=(4 5 . #0#)
Gamma> d = list(a, b);
(#0=(1 2 . #0#) #1=(4 5 . #1#))
```

Using this method, arbitrarily complex objects can be written, with all circular and self-references maintained.

As a side effect of this printing mechanism, duplicate references to objects which are not circularly defined will also be caught and correctly reproduced. For example, suppose that a list contains a single string more than once. It would be wasteful to write that string many times, and would generate an incorrect result on reading if the multiple references to that string were not preserved. The Lisp writer will correctly handle this situation:

```
Gamma> x = "Hello";
"Hello"
Gamma> a = list (x, x, x);
(#0="Hello" #0# #0#)
Gamma> eq (car(a), cadr (a));
 t
```

In the above example, if the Gamma writer did not preserve the multiple references to the string "Hello", then `a` would be printed as:

("Hello" "Hello" "Hello")

When this object is read by the Gamma reader, we would get a list which is visibly the same but for which the data references no longer match:

```
Gamma> a = list("Hello", "Hello", "Hello");
("Hello" "Hello" "Hello")
Gamma> eq (car(a), cadr(a));
nil
```

### Reading

#### Reading Gamma Expressions

Any valid Gamma expression can be read by the Gamma reader using the function `read`. The `read` function will read from the current location in a file, skipping over comments, until it encounters a character which could be the beginning of a Gamma expression. The
reader then constructs the shortest possible complete expression from the input and returns that. A complete Gamma expression may be as simple as a number, or as complex as a complete function definition or complex data object. The reader ignores white space, except as a token separator. It may be interesting to note that the entire Gamma mainline is essentially just a simple loop:

\[
\text{while } ((\text{exp} = \text{read (input\_file)}) \neq \_eof_) \text{ eval (exp);}\]

**Reading Arbitrary ASCII Data**

Gamma allows the programmer to read arbitrary ASCII data using the function `read_line`, which will read from the current file position to the first carriage return, regardless of the syntactic validity of the data on the line. If data fields are known to be separated by white space, then the `read` function using Lisp syntax may also be used to read a single field. Notice that the `read` function will treat an unquoted string of ASCII characters as a symbol, not as a string. It is more common when dealing with line-formatted data to use `read_line` followed by `string_split`.

**Reading Binary Data**

Gamma provides a number of functions for reading binary data. These functions all begin with the prefix `read_`, and they read according to the rules for C data types for the particular platform. For example, `read_char` will read a decimal representation of a string of length 1 containing a single character.
Special Topics

Modifying QNX Process Environment Variables

The QNX 4 environment variables can be read and modified by a Gamma program using the `getenv` and `setenv` function calls.

QNX 4 Interprocess Communication (IPC)

QNX 4 interprocess communication is a popular mechanism for 'talking' between software modules, based on the QNX 4 operating system. QNX 4's microkernel architecture implements message passing in such a way that only data locations are transferred between processes, making its IPC for small amounts of data as, or more, efficient than shared memory schemes. Gamma encapsulates most of the common QNX 4 IPC 'C' function calls.

Functions such as `qnx_receive` and `qnx_reply` may be redundant in a program that is using one of Gamma's built-in event-loop mechanisms. To review the built-in functionality of Gamma's event loops refer to the section on event loops.

The `qnx_name_attach` function attaches a 'name' to the current process. Names, rather than PIDs, are convenient ways to look for tasks since they are static while the PID of a program will not be.

Names are ASCII strings up to 32 character in length and can be either local or global. Local names must be unique to the node. Any attempt to register an existing local name will fail. Global names allow duplication and start with a slash '/' character. Global names are stored within a name program in QNX 4 called `nameloc`. When one process wants to look up another process's name, the `qnx_name_locate` function is called and the name to PID mapping is completed.

```
Gamma> myname = qnx_name_attach(0,"my_app");
```

The first argument to the `qnx_name_attach` function is the node on which to register the name. If the node number is zero the local node is assumed. `qnx_name_attach` returns a name id which is used with the `qnx_name_detach` function.

The `qnx_name_detach` function removes a local or global name from the local name list or the DataHub.

```
Gamma> qnx_name_detach(0,myname);
```

As with the `qnx_name_attach` function, the first argument to the `qnx_name_detach` function is the node number. The second argument must be the name id returned when attaching the name.
Once a name is registered then the qnx_name_locate function is useful for locating the task by name. The return value of this function is a dotted list of the format: (pid . copies) The pid is the process ID of the located task and copies is the number of processes that matched the name. Local names must be unique but there can be multiple instances of global names (those starting with '/').

An example of using the qnx_name_locate function follows:

Gamma> queue = qnx_name_locate(0,"qserve",0);
(91 . 1)

The PID of the qserve task is 91 and there was a single instance of that registered name found. It is important to assign the return value of the qnx_name_locate function to a variable since it is the first number in the list (PID) that is used as an argument to Gamma functions such as qnx_send, qnx_receive, qnx_reply, qnx_vc_attach, and qnx_vc_detach.

The qnx_receive function allows for the Gamma engine to remain receive-blocked on a specific PID, waiting for a message.

IMPORTANT: If Gamma is being run using a built-in event loop or using the next_event or next_event_nb functions then using the qnx_receive function MAY BE REDUNDANT. Event loops in Gamma have a built-in receive/reply mechanism.

The qnx_send function uses the QNX 4 send C/C++ function to send information between tasks. The qnx_send function is a synchronous IPC function, and as such, the sending task waits for the receiving task's reply before continuing.

The qnx_send function can be used to send Gamma expressions between Gamma modules. Gamma ships with a number of example programs, of which example 12 demonstrates the use of qnx_send to transmit and execute a function on another module. (Examples can be found in /usr/cogent/examples/ directory)

The important excerpts from this example are:

```
task = car (qnx_name_locate (0, "gui", 1000));
qnx_send (task, stringc (#Arc.fill_color = PgRGB(0xff, 0xff, 0)));
function TitleClock()
{
    win.title = date();
}
// Transmit new function
qnx_send (task, stringc (TitleClock));
// Execute new function once.
qnx_send (task, stringc (#TitleClock()));
```

The communications channel is opened by locating the task using the qnx_name_locate function and then using qnx_send. The first qnx_send sends a command for the re-
ceiving task to evaluate, in this case to change the fill color of an object named 'Arc'. The `stringc` function is used to produce an expression that is parse-able.

Next, a new function is defined, passed, and executed on the other task using two separate `qnx_send`'s.

If you are sending IPC messages to a non-Cogent IPC task the `send_string` and `send_string_async` functions should be used.

**Cogent IPC**

The Cogent IPC layer is a generalization of QNX 4's send/receive/reply IPC layer. Cogent IPC has many benefits that allow users to easily code what would be complex systems in C. Some of these services are:

- Network-wide name registration service
- Cogent DataHub exceptions and echos
- true asynchronous messages
- pseudo asynchronous messages
- synchronous messages
- QNX 4 IPC messages
- Task started notification
- Task death notification
- automatic handling of receive/reply for Cogent IPC messages
- remote procedure calls

**Cogent IPC Service Modules**

To use the Cogent IPC layer, two services optionally provided to the Gamma developer are required: `nserve` and `qserve`. These services are run as programs on the same CPU or network as Gamma.

The `nserve` command is the Cascade NameServer module. Although similar to the QNX 4 `nameloc` program in concept, this name database has some differences that make it worth using.

The `nserve` module is run on every node requiring name services. Every `nserve` module is updated on an event-basis, rather than on a timed basis as QNX 4's `nameloc` is, and therefore discrepancies between multiple `nserve`'s on a network are rare.

The `qserve` program is the asynchronous queue manager for Cogent IPC. Queues are used in Cogent IPC to implement asynchronous communication channels between two programs. The `qserve` module is run on every node requiring Cogent queue services.
Cogent IPC Advanced services

The Cogent IPC layer provides many advanced services that augment the basic send/receive/reply protocol. This section describes those services.

Cogent IPC Messages

The Cogent IPC layer provides a messaging protocol that is easier to use and different in format from raw QNX 4 send/receive/reply.

Messages between Cogent IPC-enabled tasks are very similar to function calls. A message is constructed and sent, and the task on the other end evaluates the message. The return value of the evaluation of the message is transmitted to the originating task in the reply.

Consider two Gamma modules using the following code:

Task A:

```gamma
#!/usr/cogent/bin/gamma
init_ipc("task_a");

while (t)
{
    next_event();
}
```

The function `init_ipc` is called first to initialize Cogent interprocess communication. For more details, see **IPC Initialization** below.

Task B:

```gamma
#!/usr/cogent/bin/gamma
init_ipc("task_b");

function ask_taska_date ()
{
    local result,tp;
    if (tp = locate_task("task_a",nil))
        result = send(tp,#date());
    else
        result = "could not locate task A";
}

every(1.0,#princ(ask_taska_date(),"\n");

while (t)
{
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
Of specific note in this example is the format of the message in the `send` function. The first argument to the Cogent IPC function `send` is a task. The `locate_task` function, along with the `nservice` module provides the name lookup. The second argument is an expression for the receiver to evaluate. For simple `send`'s an unevaluated Gamma expression (using `#`) will suffice. For more complex `send`'s, such as when a partially evaluated list of arguments need to be passed, the format of the `send` command should be Lisp.

This code gives a good example of using the Cogent IPC layer as an RPC (Remote Procedure Call) mechanism.

To use the Cogent IPC layer for transferring data between tasks, use the Lisp expression for assignment: `setq`. An example is:

**Task C:**

```gamma
#!/usr/cogent/bin/gamma
init_ipc("task_c");

add_set_function(#x,#princ("Task C reports x="#x,"\n"));

while (t)
{
   next_event();
}
```

**Task D:**

```gamma
#!/usr/cogent/bin/gamma
init_ipc("task_d");

function inc_x ()
{
   local result,tp;
   x++;
   if (tp = locate_task("task_c",nil))
      result = send(tp,list(#setq, #x, x));
}

x = 0;
every(0.1,#inc_x());

while (t)
{
```
In this example task C sets up a `set_function` before starting its event loop. The set function will print out the value of x if it changes. Task D initializes x to 0 and then starts a timer to run every tenth of a second to increment x and send `setq` expressions to task C.

```lisp
(setq x 1)
(setq x 2)
(setq x 3)
(setq x 4)
```

These expressions are in Lisp format because all messages between processes use the Lisp internal representation for efficiency.

The `setq` function is evaluated in task C. Any side effects of the function, for example the setting of the variable x, happens in task C. The return value of the function is the content of the reply message. The return value of the `send` function can be found by evaluating the 'result' variable in the `inc_x` function.

Consider the `inc_x` function re-written as:

```c
function inc_x ()
{
    local result,tp;
    x++;
    if (tp = locate_task("task_c",nil))
    {
        result = send(tp,list(#setq, #x, x));
        princ("task D result of send: ",result,"\n");
    }
}
```

When this example is run the return value of the send is shown to be the result of the `setq` function. Obviously, task D must wait for task C to receive and evaluate the message before sending back the response.

### Asynchronous Messages

Consider two tasks that wish to communicate: task E and task F. Task E is a time sensitive task that needs to deliver a package of data to task F. Task E cannot take the chance that task F will accept its data immediately and issue a reply so that it may continue with its own jobs. In short, a synchronous send compromises task E's job because it must wait for task F to respond before proceeding.
To send data asynchronously from task E to task F, a queue is used. Data is sent from task E to the queue. The queue responds immediately to task E, freeing it up to continue. Then a proxy, a special non-blocking message, is sent from the queue to task F. Upon receipt of the proxy, task F knows that the queue contains data for it. When task F is ready it asks the queue for the data.

With some small changes, the example from the previous section can be changed from synchronous messaging to asynchronous, as follows:

Task E:

```gamma
#!/usr/cogent/bin/gamma
init_ipc("task_e","task_e_q");
add_set_function(#x,#princ("Task E reports x=",x,"\n");
while (t)
{
    next_event();
}
```

Task F:

```gamma
#!/usr/cogent/bin/gamma
init_ipc("task_f","task_f_q");
function inc_x ()
{
    local result,tp;
    x++;
    if (tp = locate_task("task_e",nil))
    {
        result = send_async(tp,list(#setq, #x, x));
        princ("task F result of send: ",result,"\n");
    }
}
x = 0;
every(0.1,#inc_x());
while (t)
{
    next_event();
}
```
The `init_ipc` function calls at the beginning of each module now open a queue name with `qserve`, and the `inc_x` function has been changed to use `send_async` instead of `send`.

When this example is run the results show that task F receives a `t` (true) that the message was delivered but does not have to wait for task E to generate the result of the expression.

Using asynchronous communication immediately solves the dead-lock problem that all developers of multi-module systems must eventually face. To the developer, the use of asynchronous communication in Gamma entails only the use of a slightly different function: `send_async` instead of `send`.

**Pseudo-Asynchronous Messages**

For situations where the `qserve` program is not running and an asynchronous non-blocking IPC call is required then Gamma pseudo-asynchronous IPC call can be used.

The `isend` function sends a message between two Cogent IPC enabled tasks. Immediately upon receipt of the message, the receiver replies that the message was received. The return value of the received message is not sent back.

**Task Started & Death Notification**

When a task registers a name with `nserv` it can thereafter receive information regarding any other `nserv` registered task that starts or stops.

This is done by defining two functions with specific names, each within their respective code, to handle this information. The functions are:

```c
function taskstarted_hook (name, queue, domain, node, id);
```

and

```c
function taskdied_hook (name, queue, domain, node, id);
```

The body of each of these functions is up to the programmer. Most "hook" functions check the name, queue, and possibly the domain of the started/stopped task and then take a specific action such as:

- restarting a task that has died;
- informing the user that a module has died;
- inform other modules that a new service is available;
- query the new module for information; and,
- Cogent DataHub start/stop.

**Automatic Handling of QNX 4 receive and reply**

The following Gamma functions automatically handle QNX 4 `receive/reply`:
• PtMainLoop
• next_event
• next_event_nb
• flush_events

IPC Initialization

Before any form of Cogent interprocess communication occurs there must be a call to the `init_ipc` function. This function opens the channels of communications between Gamma and other tasks powered by Gamma, Cascade Connect, or other Cogent products. With this function you determine your task's name and optionally its queue name and domain.

A program's name is the string registered with the nserve program. Gamma names and queue names for tasks should be unique on the network. A program’s queue name is the name of the queue that is registered if it wants to participate in asynchronous communication using Cogent's qserve utilities. The domain name is the name of the default Cogent DataHub domain from which to read and write points.

It is typical to find the `init_ipc` function called within the first few calls in the program. Here's an example:

```bash
#!/usr/local/bin/gamma
require_lisp("PhotonWidget");
require_lisp("PhabTemplate");

mynname = car(argv);
init_ipc("mynname");
```

This program segment first defines the engine to run on the first line, then loads some required files for Photon widget manipulation and Photon Application Builder support. The `argv` variable holds the arguments passed to Gamma. The first item in the list is the name of the executable, which is put in the `mynname` variable. The `init_ipc` function is then called with the registered name being whatever the name of the program happens to be.

Locating Tasks

Using Gamma's IPC communications protocol, a task can be located by name or by id. This protocol allows for synchronous, asynchronous, and semi-asynchronous communications between Gamma, SCADALisp, and other Cogent products such as Cascade Connect and the Cogent DataHub.

Locating a task by name can be done with the `locate_task` function. This is similar to using the `qnx_name_locate` function except that, since nserve's names are intended to be unique on a network the node number need not be specified.

```bash
marko:/home/marko$ gamma -q
```
The return value of the `locate_task` function is a Gamma task type. The task type is an internal representation of the task that was located. There is nothing the user can do with variables of this data type other than to pass them through as arguments to Cogent IPC functions.

To locate a task on a specific node with a specific PID number use the `locate_task_id` function.

Before using either `locate_task` or `locate_task_id`, the `init_ipc` function must have already been called.

Once discussions with a task are completed, the channel should be closed using the `close_task` function.

### Transmitting Character Strings

The `send_string` and `send_string_async` functions are used to format a message to be sent to a non-Cogent IPC task. These functions will accept a string (text surrounded by quotes) as a parameter, and will send the contents of the string without the enclosing quotes. Note that the normal `send` function will send the enclosing quotes as part of the message.

### Cogent DataHub

The Cogent DataHub is a high performance data collection and distribution center designed for easy integration with a Gamma application. Just as QNX 4 is an excellent choice for developers of systems that must acquire real-time data, the Cogent DataHub is the right choice for distribution of that data.

The Cogent DataHub provides:

- data services to its clients by exception and lookup;
- asynchronous data delivery ensuring client task protection blocking;
- network connection/reconnection issues;
- data services to many clients at once;
- transparent data services to/from Gamma;
- flexible data tag names;
- inherent understanding of data types (as Gamma does);
- time-stamping of data;
- C libraries for the creation of custom clients;
• security access levels on data points; and,
• a confidence value for assigning fuzzy values to data points.

The Cogent DataHub is:
• a convenient way to disseminate real-time data;
• a RAM resident module holding current data;
• a proven solution with thousands of hours of installed performance; and,
• a great source of information for:
  • historical & relational database;
  • hard disk loggers; and,
  • Cascade Connect real-time connection to MS-Windows.

The Cogent DataHub is not:
• a historical database;
• a relational database;
• a hard disk logger;
• slow;
• a large memory requirement module; or,
• pre-configured.

Whenever multiple tasks are communicating there is a chance for a deadlock situation. The Cogent DataHub is at the center of many mission critical applications because it provides real-time data to its clients without the threat of being blocked on the receiving task. The Cogent DataHub never blocks on a task that is busy. The DataHub is always able to receive data from clients because it uses the qserve manager to handle outgoing messages. The DataHub only ever sends messages to the Cascade QueueServer program, which is optimized to never enter a state where it cannot accept a message from the Cogent DataHub.

**Cogent DataHub Exceptions and Echos**

When a new data point is sent to the Cogent DataHub the DataHub automatically updates its clients that are interested in the point. Some clients get information from the the DataHub on request only, by polling. Other clients register with the Cogent DataHub for changes in some or all points, called *exceptions*.

The Cogent DataHub not only allows its clients to register and receive exceptions on data points, but also provides a special message type called an *echo* that is extremely important in multi-node or multi-task applications.

When the Cogent DataHub receives a new data point it immediately informs its registered clients of the new data value. The clients will receive an asynchronous exception message. In some circumstances, the client that sent the new data value to the DataHub is al-
so registered for an exception on that point. In this case, the originator of the data change will also receive an exception indicating the data change. When there are multiple clients reading and writing the same data point a client may wish to perform an action whenever another client changes the data. Thus, it must be able to differentiate between exceptions which it has originated itself, and ones which originate from other clients. The Cogent DataHub defines an echo as an exception being returned to the originator of the value change.

In certain circumstances, the lack of differentiation between exceptions and echos can introduce instability into both single and multi-client systems. For example, consider an application that communicates with another Lisp or MMI system, such as Wonderware’s InTouch. InTouch communicates via DDE, which does not make the distinction between exceptions and echos. A data value delivered to InTouch will always be re-emitted to the sender, which will cause the application to re-emit the value to the Cogent DataHub. The DataHub will generate an exception back to the application, which will pass this to InTouch, which will re-emit the value to the application, which will send it to the DataHub, on so on. A single value change will cause an infinite communication loop. There are many other instances of this kind of behavior in asynchronous systems. By introducing echo capability into the DataHub, the cycle is broken immediately because the application can recognize that it should not re-emit a data change that it originated itself.

The echo facility is necessary for another reason. It is not sufficient to simply not emit the echo to the originating task. If two tasks read and write a single data point to the DataHub, then the DataHub and both tasks must still agree on the most recent value. When both tasks attempt to write the point, one gets an exception and updates its current value to agree with the DataHub and the sender. If both tasks simultaneously emit different values, then the task whose message is processed first will get an exception from the first, and the first will get an exception from the second. In effect, the two tasks will swap values, and only one will agree with the DataHub. The echo message solves this dilemma by allowing the task whose message was processed second to receive its own echo, causing it to realize that it had overwritten the exception from the other task.
## Appendix A. Function List

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute_path</td>
<td>returns the absolute path of the given file.</td>
</tr>
<tr>
<td>access</td>
<td>checks a file for various permissions.</td>
</tr>
<tr>
<td>acos</td>
<td>finds the arc cosine of a number.</td>
</tr>
<tr>
<td>add_echo_function</td>
<td>assigns functions for echoes on a point.</td>
</tr>
<tr>
<td>add_exception_function</td>
<td>assigns functions for exceptions on a point.</td>
</tr>
<tr>
<td>add_hook</td>
<td>hooks a function to an event.</td>
</tr>
<tr>
<td>add_set_function</td>
<td>sets an expression to be evaluated when a given symbol changes value.</td>
</tr>
<tr>
<td>after</td>
<td>performs an action after a period of time.</td>
</tr>
<tr>
<td>alist_p</td>
<td>tests for association lists.</td>
</tr>
<tr>
<td>allocated_cells</td>
<td>gives the number of allocated and free cells.</td>
</tr>
<tr>
<td>and</td>
<td>is the same as the corresponding logical operator (&amp;&amp;).</td>
</tr>
<tr>
<td>append</td>
<td>concatenates several lists into a single new list.</td>
</tr>
<tr>
<td>apropos</td>
<td>finds all defined symbols in the current interpreter environment.</td>
</tr>
<tr>
<td>aref</td>
<td>returns an expression at a given index.</td>
</tr>
<tr>
<td>array</td>
<td>constructs an array.</td>
</tr>
<tr>
<td>array_p</td>
<td>tests for arrays.</td>
</tr>
<tr>
<td>array_to_list</td>
<td>converts an array to a list.</td>
</tr>
<tr>
<td>aset</td>
<td>sets an array element to a value at a given index.</td>
</tr>
<tr>
<td>asin</td>
<td>finds the arc sine of a number.</td>
</tr>
<tr>
<td>assoc</td>
<td>searches an association list for a sublist, using eq.</td>
</tr>
<tr>
<td>assoc_equal</td>
<td>searches an association list for a sublist, using equal.</td>
</tr>
<tr>
<td>at</td>
<td>performs an action at a given time, or regularly.</td>
</tr>
<tr>
<td>atan</td>
<td>finds the arc tangent of a number.</td>
</tr>
<tr>
<td>atan2</td>
<td>finds the arc tangent with two arguments.</td>
</tr>
<tr>
<td>atexit</td>
<td>evaluates code before exiting a program.</td>
</tr>
<tr>
<td>AutoLoad</td>
<td>allows for run-time symbol lookup.</td>
</tr>
<tr>
<td>autoload Undefined Symbol</td>
<td>checks undefined symbols for AutoLoad.</td>
</tr>
<tr>
<td>AutoMapFunction</td>
<td>maps a C function to a Gamma function.</td>
</tr>
<tr>
<td>autotrace_p</td>
<td>is for internal use only.</td>
</tr>
<tr>
<td>backquote</td>
<td>corresponds to a quote operator.</td>
</tr>
<tr>
<td>band</td>
<td>performs bitwise and operations.</td>
</tr>
<tr>
<td>basename</td>
<td>gives the base of a filename.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>bdelete</code></td>
<td>deletes a single character from a buffer.</td>
</tr>
<tr>
<td><code>bin</code></td>
<td>converts numbers into binary form.</td>
</tr>
<tr>
<td><code>binsert</code></td>
<td>inserts a value into a buffer.</td>
</tr>
<tr>
<td><code>block_signal</code></td>
<td>starts signal blocking.</td>
</tr>
<tr>
<td><code>block_timers</code></td>
<td>blocks timer firing.</td>
</tr>
<tr>
<td><code>bnot</code></td>
<td>performs bitwise not operations.</td>
</tr>
<tr>
<td><code>bor</code></td>
<td>performs bitwise inclusive or operations.</td>
</tr>
<tr>
<td><code>breakpoint_p</code></td>
<td>is for internal use only.</td>
</tr>
<tr>
<td><code>bsearch</code></td>
<td>searches an array or list for an element.</td>
</tr>
<tr>
<td><code>buffer</code></td>
<td>constructs a buffer.</td>
</tr>
<tr>
<td><code>buffer_p</code></td>
<td>tests for buffers.</td>
</tr>
<tr>
<td><code>buffer_to_string</code></td>
<td>converts a buffer to a string.</td>
</tr>
<tr>
<td><code>builtin_p</code></td>
<td>is for internal use only.</td>
</tr>
<tr>
<td><code>bxor</code></td>
<td>perform bitwise exclusive or operations.</td>
</tr>
<tr>
<td><code>caaar</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>caadr</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>caar</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cadar</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>caddr</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cadr</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>call</code></td>
<td>calls a class method for a given instance.</td>
</tr>
<tr>
<td><code>cancel</code></td>
<td>removes a timer from the set of pending timers.</td>
</tr>
<tr>
<td><code>car</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cd</code></td>
<td>changes the working directory.</td>
</tr>
<tr>
<td><code>cdaar</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cdadr</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cdar</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cddar</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cdddr</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cddr</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>cdr</code></td>
<td>returns that element of a list.</td>
</tr>
<tr>
<td><code>ceil</code></td>
<td>rounds a real number up to the next integer.</td>
</tr>
<tr>
<td><code>cfand</code></td>
<td>performs and operations with a confidence factor.</td>
</tr>
<tr>
<td><code>cfor</code></td>
<td>performs or operations with a confidence factor.</td>
</tr>
<tr>
<td><code>char</code></td>
<td>generates an ASCII character from a number.</td>
</tr>
</tbody>
</table>
Function List

char_val generates a character's numeric value.
chars_waiting checks for characters waiting to be read on a file.
class_add_cvar adds new class variables.
class_add_ivar adds an instance variable to a class.
class_name gives the name of the class.
class_of gives the class definition of a given instance.
class_p tests for classes.
ClearAutoLoad removes all AutoLoad rules.
clock gets the OS time.
close closes an open file.
close_task closes a task opened by locate_task.
conf queries confidence factors.
cons constructs a cons cell.
cons_p tests for cons cells.
constant_p tests for constants.
copy makes a copy of the top list level of a list.
copy_tree copies the entire tree structure and elements of a list.
cos returns the cosine of a number.
create_state is part of the SCADAlisp exception-driven state machine mechanism.
date gets the OS date and time; translates seconds into dates.
date_of is obsolete, see date
dec converts numbers into base-10 form.
defclass is the function equivalent of the statement: class.
defmacro is a Lisp equivalent of the function: macro.
defmacroe is a Lisp equivalent of the function: macro.
defmethod is the function equivalent of the function: method.
defun is a function equivalent of the statement: function.
defune is a function equivalent of the statement: function.
defvar defines a global variable with an initial value.
delete removes an element from an array.
destroy destroys a class instance.
destroyed_p tests for destroyed instances.
_destroy_task should never be used.
difference constructs a list of the differences between two lists.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory</td>
<td>returns the contents of a directory.</td>
</tr>
<tr>
<td>dirname</td>
<td>returns the directory path of a file.</td>
</tr>
<tr>
<td>div</td>
<td>divides two numbers, giving an integer result.</td>
</tr>
<tr>
<td>dlclose</td>
<td>closes an open dynamic library.</td>
</tr>
<tr>
<td>derror</td>
<td>reports errors in dl functions.</td>
</tr>
<tr>
<td>dlfunc</td>
<td>reserved for future use.</td>
</tr>
<tr>
<td>DllLoad</td>
<td>loads dynamic libraries.</td>
</tr>
<tr>
<td>dimethod</td>
<td>reserved for future use.</td>
</tr>
<tr>
<td>dlopen</td>
<td>loads a dynamic library from a file.</td>
</tr>
<tr>
<td>drain</td>
<td>modifies end-of-file detection.</td>
</tr>
<tr>
<td>enter_state</td>
<td>is part of the SCADALisp exception-driven state machine mechanism.</td>
</tr>
<tr>
<td>eq</td>
<td>compares for identity and equivalence.</td>
</tr>
<tr>
<td>equal</td>
<td>compares for identity and equivalence.</td>
</tr>
<tr>
<td>errno</td>
<td>detects and numbers errors.</td>
</tr>
<tr>
<td>error</td>
<td>redirects the interpreter.</td>
</tr>
<tr>
<td>eval</td>
<td>evaluates an argument.</td>
</tr>
<tr>
<td>eval_count</td>
<td>counts evaluations made since a program started.</td>
</tr>
<tr>
<td>eval_list</td>
<td>evaluates each element of a list.</td>
</tr>
<tr>
<td>eval_string</td>
<td>evaluates a string.</td>
</tr>
<tr>
<td>every</td>
<td>performs an action every number of seconds.</td>
</tr>
<tr>
<td>exec</td>
<td>executes a program.</td>
</tr>
<tr>
<td>exit_program</td>
<td>terminates the interpreter.</td>
</tr>
<tr>
<td>exit_state</td>
<td>is part of the SCADALisp exception-driven state machine mechanism.</td>
</tr>
<tr>
<td>exp</td>
<td>calculates an exponent of the logarithmic base (e).</td>
</tr>
<tr>
<td>fd_close</td>
<td>closes a open file identified by a file descriptor.</td>
</tr>
<tr>
<td>fd_data_function</td>
<td>attaches a write-activated callback to a file.</td>
</tr>
<tr>
<td>fd_eof_function</td>
<td>attaches an <em>eof</em>-activated callback to a file.</td>
</tr>
<tr>
<td>fd_open</td>
<td>opens a file or device and assigns it a file descriptor.</td>
</tr>
<tr>
<td>fd_read</td>
<td>reads a buffer or string from an open file identified by a file descriptor.</td>
</tr>
<tr>
<td>fd_to_file</td>
<td>creates a file pointer from a descriptor.</td>
</tr>
<tr>
<td>fd_write</td>
<td>writes a buffer or string to an open file identified by a file descriptor.</td>
</tr>
<tr>
<td>file_date</td>
<td>gives the file modification date.</td>
</tr>
</tbody>
</table>
file_p  tests for files.
file_size  gives the file size.
fileno  creates a file descriptor from a pointer.
find  searches a list using the function: eq.
find_equal  searches a list using the function: equal.
fixed_point_p  tests for fixed-point reals.
floor  rounds a real number down to its integer value.
flush  flushes any pending output on a file or string.
flush_events  handles all pending events, then exits.
fork  duplicates a process.
format  generates a formatted string.
free_cells  returns the number of available memory cells.
funcall  provides compatibility with other Lisp dialects.
function_args  lists the arguments of a function.
function_body  gives the body of a user-defined function.
function_calls  tells how often a function was called during profiling.
function_name  gives the name of a function.
function_p  tests for functions.
function_runtime  gives the time a function has run during profiling.
gc  runs the garbage collector.
gc_blocksize  is for internal use only.
gc_enable  is for internal use only.
gc_newblock  is for internal use only.
gc_trace  controls the tracing of garbage collection.
gensym  generates a unique symbol.
getcwd  gets the current working directory.
getenv  retrieves the value of an environment variable.
gethostname  gets the computer's host name.
getnid  returns the local node number.
getpid  returns the program ID.
getprop  returns a property value for a symbol.
getsockopt  gets a socket option.
has_cvar  queries for the existence of a class variable.
has_ivar  queries for the existence of an instance variable.
hex  converts numbers into hexadecimal form.
init_async_ipc requests queue information from a task.
init_ipc sets up necessary data structures for IPC.
insert inserts a value at a given position.
instance_p tests for instances.
instance_vars finds all the instance variables of a class or instance.
int converts numbers to integer form.
int_p tests for integers.
intersection constructs a list of all the elements found in both of two lists.
ioctl performs control functions on a file descriptor.
is_busy determines if a file is busy.
is_class_member checks if an instance or class is a member of a class.
is_dir determines if a file is a directory.
is_file determines if a file exists.
is_readable determines if a file is readable.
is_writable determines if a file is writable.
isend sends a synchronous message and doesn't wait for the result.
ivar_type returns the type of a given instance variable.
nkill sends a signal to a process.
length counts the number of elements in a list or array.
list creates lists, evaluating the arguments.
list_p tests for lists.
list_to_array converts a list to an array.
listq creates lists without evaluating the arguments.
load loads files.
load_lisp loads Lisp files.
locate_task finds and connects to tasks by name.
locate_task_id finds and connects to tasks by task ID and network node.
lock_point locks or unlocks points.
log calculates natural logarithms.
log10 calculates base 10 logarithms.
logn calculates logarithms of a given base.
long_p tests for long integers.
macro helps generate custom functions.
macro_p  tests for macros.
make_array  creates an empty array.
make_buffer  creates a new, empty buffer.
method_p  tests for methods.
mkdir  creates a new sub-directory.
modules  is obsolete, and returns nothing of value.
name_attach  attaches a name to a task.
nanoclock  gets the OS time, including nanoseconds.
nanosleep  pauses the interpreter for seconds and nanoseconds.
nappend  appends one or more lists, destructively modifying them.
neg  negates a number.
new  creates a new instance of a class.
next_event  blocks waiting for an event, and calls the event handling function.
next_event_nb  is the same as next_event, but doesn't block.
nil_p  tests for nil values.
NoAutoLoad  removes selected AutoLoad rules.
not  is the same as the corresponding logical operator (!).
notrace  turns tracing off.
rremove  removes list items, destructively altering the list.
nreplace  replaces elements in a list, using eq.
nreplace_equal  replaces elements in a list, using equal.
nserve_query  puts information from nserve into an array.
nth_car  iteratively applies the car functions to a list.
nth_cdr  iteratively applies the cdr functions to a list.
number  attempts to convert an expression to a number.
number_p  tests for numbers.
oct  converts numbers into octal form.
open  attempts to open a file.
open_string  allows a string to be used as a file.
or  is the same as the corresponding logical operator (||).
parent_class  returns the closest parent (base) of a class or instance.
parse_string  parses an input string.
path_node  gives the node number of a path in a QNX 2 path definition.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipe</td>
<td>creates a pipe.</td>
</tr>
<tr>
<td>point_locked</td>
<td>indicates if a point is locked.</td>
</tr>
<tr>
<td>point_nanoseconds</td>
<td>gives the nanoseconds from point_seconds that a point value changed.</td>
</tr>
<tr>
<td>point_seconds</td>
<td>gives the time the point value changed.</td>
</tr>
<tr>
<td>point_security</td>
<td>gives the security level of a point.</td>
</tr>
<tr>
<td>pow</td>
<td>raises a base to the power of an exponent.</td>
</tr>
<tr>
<td>pretty_princ</td>
<td>writes to the standard output file, with formatting.</td>
</tr>
<tr>
<td>pretty_print</td>
<td>writes Lisp-readable output to the standard output file, with formatting.</td>
</tr>
<tr>
<td>pretty_write</td>
<td>writes an expression to a file, applying formatting.</td>
</tr>
<tr>
<td>pretty_writec</td>
<td>writes an expression to a file, applying formatting.</td>
</tr>
<tr>
<td>princ</td>
<td>writes to the standard output file.</td>
</tr>
<tr>
<td>print</td>
<td>writes Lisp-readable output to the standard output file.</td>
</tr>
<tr>
<td>print_stack</td>
<td>prints a Gamma stack.</td>
</tr>
<tr>
<td>profile</td>
<td>collects statistics on function usage and run time.</td>
</tr>
<tr>
<td>progl</td>
<td>groups several statements into one expression.</td>
</tr>
<tr>
<td>progn</td>
<td>groups several statements into one expression.</td>
</tr>
<tr>
<td>properties</td>
<td>should never be used.</td>
</tr>
<tr>
<td>pty</td>
<td>runs programs in a pseudo-tty.</td>
</tr>
<tr>
<td>ptytio</td>
<td>runs programs in a pseudo-tty, using a termios structure argument.</td>
</tr>
<tr>
<td>quote</td>
<td>corresponds to a quote operator.</td>
</tr>
<tr>
<td>random</td>
<td>generates random numbers from 0 to 1.</td>
</tr>
<tr>
<td>raw_memory</td>
<td>tells the amount of memory in use.</td>
</tr>
<tr>
<td>read</td>
<td>reads a Lisp expression from a file.</td>
</tr>
<tr>
<td>read_char</td>
<td>reads the next character from the input file.</td>
</tr>
<tr>
<td>read_double</td>
<td>reads the next double from the input file.</td>
</tr>
<tr>
<td>read_eval_file</td>
<td>reads a file, evaluating and counting expressions.</td>
</tr>
<tr>
<td>read_existing_point</td>
<td>retrieves points.</td>
</tr>
<tr>
<td>read_float</td>
<td>reads the next float from the input file.</td>
</tr>
<tr>
<td>read_line</td>
<td>reads a single line of text.</td>
</tr>
<tr>
<td>read_long</td>
<td>reads the next long value from the input file.</td>
</tr>
<tr>
<td>read_n_chars</td>
<td>reads and stores characters.</td>
</tr>
<tr>
<td>read_point</td>
<td>creates and/or retrieves points.</td>
</tr>
<tr>
<td>read_short</td>
<td>reads the next short value from the input file.</td>
</tr>
</tbody>
</table>
Function List

read_until reads characters, constructing a string as it goes.
real_p tests for reals.
register_all_points registers an application to receive exceptions for all points.
register_exception is not yet documented.
register_existing_point registers an application to receive exceptions for a single existing point.
register_point creates and/or registers an application to receive exceptions for a single point.
registered_p tests for registered points.
remove removes list items without altering the list.
remove_echo_function removes an echo function from a symbol.
remove_exception_function removes an exception function from a symbol.
remove_hook removes a hooked function.
remove_set_function removes a set function from a symbol.
rename renames a file.
require requires/loads files.
require_lisp requires/loads Lisp files.
required_file determines which files would be loaded.
reverse reverses the order of list elements.
root_path strips the final file or directory name from a path.
round rounds a real number up or down to the nearest integer.
rplaca replaces the car of a list.
rplacd replaces the cdr of a list.
runt_hooks runs a hooked function.
secure_point alters the security level on a point.
seek sets the file position for reading or writing.
send transmits expressions for evaluation.
send_async transmits expressions asynchronously.
send_string transmits strings for evaluation.
send_string_async transmits a string asynchronously.
ser_setup sets parameters for a serial port device.
set assigns a value to a symbol, evaluating both arguments.
set_autotrace is reserved for future use.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_breakpoint</td>
<td>is reserved for future use.</td>
</tr>
<tr>
<td>set_conf</td>
<td>sets confidence factors.</td>
</tr>
<tr>
<td>set_domain</td>
<td>sets the default domain for future calls.</td>
</tr>
<tr>
<td>set_random</td>
<td>starts random at a different initial number.</td>
</tr>
<tr>
<td>set_security</td>
<td>changes the security level for the current process.</td>
</tr>
<tr>
<td>setenv</td>
<td>sets an environment variable for the current process.</td>
</tr>
<tr>
<td>setprop</td>
<td>sets a property value for a symbol.</td>
</tr>
<tr>
<td>setprops</td>
<td>lists the most recent property value settings.</td>
</tr>
<tr>
<td>setq</td>
<td>assigns a value to a symbol, evaluating the second argument.</td>
</tr>
<tr>
<td>setqq</td>
<td>assigns a value to a symbol, not evaluating any arguments.</td>
</tr>
<tr>
<td>setsockopt</td>
<td>sets a socket option.</td>
</tr>
<tr>
<td>shell_match</td>
<td>compares string text to a pattern.</td>
</tr>
<tr>
<td>shm_open</td>
<td>opens shared memory objects.</td>
</tr>
<tr>
<td>shm_unlink</td>
<td>removes shared memory objects.</td>
</tr>
<tr>
<td>shorten_array</td>
<td>reduces or expands the size of an array.</td>
</tr>
<tr>
<td>shorten_buffer</td>
<td>reduces the size of a buffer.</td>
</tr>
<tr>
<td>signal</td>
<td>defines an expression to be evaluated at an OS-generated signal.</td>
</tr>
<tr>
<td>sin</td>
<td>returns the sine of a number.</td>
</tr>
<tr>
<td>sleep</td>
<td>suspends execution for seconds.</td>
</tr>
<tr>
<td>sort</td>
<td>sorts a list or array, destructively modifying the order.</td>
</tr>
<tr>
<td>sqr</td>
<td>finds the square of a number.</td>
</tr>
<tr>
<td>sqrt</td>
<td>finds the square root of a number.</td>
</tr>
<tr>
<td>stack</td>
<td>lists all functions called so far.</td>
</tr>
<tr>
<td>strchr</td>
<td>searches a string for a character, returning the first location.</td>
</tr>
<tr>
<td>strcmp</td>
<td>compares strings, case-sensitive.</td>
</tr>
<tr>
<td>strerror</td>
<td>retrieves an error message.</td>
</tr>
<tr>
<td>strcmp</td>
<td>compares strings, case-insensitive.</td>
</tr>
<tr>
<td>string</td>
<td>constructs a string.</td>
</tr>
<tr>
<td>string_file_buffer</td>
<td>queries a string file for its internal buffer.</td>
</tr>
<tr>
<td>string_p</td>
<td>tests for strings.</td>
</tr>
<tr>
<td>string_split</td>
<td>breaks a string into individual words.</td>
</tr>
<tr>
<td>string_to_buffer</td>
<td>creates a buffer object from a string.</td>
</tr>
</tbody>
</table>
stringc constructs a string in Lisp-readable form,
strlen counts the number of characters in a string.
strncmp compares two strings and return a numeric result, case-sensitive.
strnicmp compares two strings and return a numeric result, case-insensitive.
strrchr searches a string for a character, returning the last location.
strrev reverses the order of characters in a string.
strstr finds the location of a given substring.
substr returns a substring for a given location.
sym-alist_p tests for symbolic association lists.
symbol constructs a symbol from a string.
symbol_p tests for symbols.
system treats its argument as a system command.
tan returns the tangent of a number.
taskdied calls a function when a task stops.
task_info gets information from a task descriptor.
taskstarted calls a function when a task starts.
tell indicates file position.
terpri prints a newline to an open file.
time gives command execution times.
timer_is_proxy controls timer handling in Gamma.
tmpfile generates temporary output file names.
tolower converts upper case letters to lower case.
toupper converts lower case letters to upper case.
trace turns tracing on.
trap_error traps errors in the body code.
true_p tests for truth value.
unblock_signal ends signal blocking.
unblock_timers unblocks timer firing.
unbuffer_file causes a file to be treated as unbuffered on both input and output.
undefined_p tests for undefined values.
undefined_symbol_p tests for undefined symbols.
union constructs a list containing all the elements of two lists.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>unlink</strong></td>
<td>deletes a file.</td>
</tr>
<tr>
<td><strong>unread_char</strong></td>
<td>attempts to replace a character to a file for subsequent reading.</td>
</tr>
<tr>
<td><strong>unregister_point</strong></td>
<td>stops echo and exception message sending.</td>
</tr>
<tr>
<td><strong>unwind_protect</strong></td>
<td>ensures code will be evaluated, despite errors in the body code.</td>
</tr>
<tr>
<td><strong>usleep</strong></td>
<td>suspends execution for microseconds.</td>
</tr>
<tr>
<td><strong>wait</strong></td>
<td>waits for process exit status.</td>
</tr>
<tr>
<td><strong>when_echo_fns</strong></td>
<td>indicates the functions for echos on a point.</td>
</tr>
<tr>
<td><strong>when_exception_fns</strong></td>
<td>indicates the functions for exceptions on a point.</td>
</tr>
<tr>
<td><strong>when_set_fns</strong></td>
<td>returns all functions set for a symbol.</td>
</tr>
<tr>
<td><strong>whence</strong></td>
<td>gives input information.</td>
</tr>
<tr>
<td><strong>write</strong></td>
<td>writes an expression to a file.</td>
</tr>
<tr>
<td><strong>writec</strong></td>
<td>writes a Lisp expression to a file.</td>
</tr>
<tr>
<td><strong>write_existing_point</strong></td>
<td>writes values to existing points.</td>
</tr>
<tr>
<td><strong>write_n_chars</strong></td>
<td>writes characters from a buffer to a file.</td>
</tr>
<tr>
<td><strong>write_point</strong></td>
<td>writes point values, creating points if necessary.</td>
</tr>
</tbody>
</table>

Function List
Appendix B. GNU Lesser General Public License

This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.

Copyright © 1991, 1999 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Version 2.1, February 1999

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code; that you receive source code and can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

We protect your rights with a two-step method:
1. we copyright the library, and
2. we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients
should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the Lesser General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users’ freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a “work based on the library” and a “work that uses the library”. The former contains code derived from the library, whereas the latter must be combined with the library in order to run.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called “this License”). Each licensee is addressed as “you”.

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

Section 1

You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.
b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.
Section 4

You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

Section 5

A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

Section 6

As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.
You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

Section 7

You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:
a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

Section 8

You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 9

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

Section 10

Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

Section 11

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other proper-
ty right claims or to contest validity of any such claims; this section has the sole purpose
of protecting the integrity of the free software distribution system which is implemented
by public license practices. Many people have made generous contributions to the wide
range of software distributed through that system in reliance on consistent application of
that system; it is up to the author/donor to decide if he or she is willing to distribute soft-
ware through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of
the rest of this License.

Section 12

If the distribution and/or use of the Library is restricted in certain countries either by
patents or by copyrighted interfaces, the original copyright holder who places the Library
under this License may add an explicit geographical distribution limitation excluding those
countries, so that distribution is permitted only in or among countries not thus exclud-
ed. In such case, this License incorporates the limitation as if written in the body of this Li-
cense.

Section 13

The Free Software Foundation may publish revised and/or new versions of the Lesser
General Public License from time to time. Such new versions will be similar in spirit to the
present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version
number of this License which applies to it and “any later version”, you have the option of
following the terms and conditions either of that version or of any later version published
by the Free Software Foundation. If the Library does not specify a license version number,
you may choose any version ever published by the Free Software Foundation.

Section 14

If you wish to incorporate parts of the Library into other free programs whose distribu-
tion conditions are incompatible with these, write to the author to ask for permission. For
software which is copyrighted by the Free Software Foundation, write to the Free Software
Foundation; we sometimes make exceptions for this. Our decision will be guided by the
two goals of preserving the free status of all derivatives of our free software and of pro-
moting the sharing and reuse of software generally.

NO WARRANTY Section 15

BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE
LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE
STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE
LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, IN-
CLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PER-
FORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU
ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 16

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, IN-
CLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO
LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU
OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFT-
WARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY
OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the pub-
ic, we recommend making it free software that everyone can redistribute and change.
You can do so by permitting redistribution under these terms (or, alternatively, under the
terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them
to the start of each source file to most effectively convey the exclusion of warranty; and
each file should have at least the “copyright” line and a pointer to where the full notice is
found.

###lt;one line to give the library’s name and a brief idea of what it does.###gt; Copyright
(C) ###lt;year###gt; ###lt;name of author###gt;

This library is free software; you can redistribute it and/or modify it under the terms of the
GNU Lesser General Public License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this
library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor,
Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any,
to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the
names:
Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

###lt;signature of Ty Coon###gt;, 1 April 1990 Ty Coon, President of Vice

That’s all there is to it!
Symbols and Literals
Data Types and Predicates

— testing for data types.

Unlike many languages, just about every expression in Gamma is a data type. This gives the flexibility to manipulate functions, arrays, lists, classes, instances and so on as if they were data.

The following data types are defined in Gamma. Beside each data type is the name of a function which can be used to test an expression for that type. These functions are called predicates, and will return t if the test is true (the expression is that data type), or nil if it is false.

Table 3. Data Types and Related Predicates

<table>
<thead>
<tr>
<th>Type</th>
<th>Predicate</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>alist</td>
<td>alist_p</td>
<td>An association list. See assoc.</td>
</tr>
<tr>
<td>array</td>
<td>array_p</td>
<td>See array and Lists and Arrays.</td>
</tr>
<tr>
<td>autotrace</td>
<td>autotrace_p</td>
<td></td>
</tr>
<tr>
<td>breakpoint</td>
<td>breakpoint_p</td>
<td></td>
</tr>
<tr>
<td>buffer</td>
<td>buffer_p</td>
<td>See buffer.</td>
</tr>
<tr>
<td>builtin</td>
<td>builtin_p</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>class_p</td>
<td>See class.</td>
</tr>
<tr>
<td>cons</td>
<td>cons_p</td>
<td>See cons and list.</td>
</tr>
</tbody>
</table>
| constant               | constant_p      | Constants can be assigned or defined. See defvar and ::=.
<p>| destroyed instance     | destroyed_p     | See new (instance).           |
| file                   | file_p          | See open and open_string.    |
| fixed-point real       | fixed_point_p   | See Numeric Types.            |
| function               | function_p      | See function.                 |
| instance               | instance_p      | See new (instance).           |
| integer                | int_p, long_p   | See Literals.                 |
| list                   | list_p          | See list and Lists and Arrays.|
| macro                  | macro_p         | See macro.                    |
| method                 | method_p (obsolete, always returns nil) | See method. |</p>
<table>
<thead>
<tr>
<th>Type</th>
<th>Predicate</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>nil</td>
<td>nil_p</td>
<td>See nil.</td>
</tr>
<tr>
<td>number</td>
<td>number_p</td>
<td>Integer and floating point values are both considered numbers. See Literals.</td>
</tr>
<tr>
<td>real</td>
<td>real_p</td>
<td>See Literals.</td>
</tr>
<tr>
<td>registered</td>
<td>registered_p</td>
<td>See register_point.</td>
</tr>
<tr>
<td>string</td>
<td>string_p</td>
<td>See Literals and string.</td>
</tr>
<tr>
<td>sym-alist</td>
<td>sym_alist_p</td>
<td>A symbol-indexed association list. See assoc.</td>
</tr>
<tr>
<td>symbol</td>
<td>symbol_p</td>
<td>See Literals.</td>
</tr>
<tr>
<td>t</td>
<td>true_p</td>
<td>See t.</td>
</tr>
<tr>
<td>task descriptor</td>
<td>none</td>
<td>See locate_task.</td>
</tr>
<tr>
<td>undefined</td>
<td>undefined_p</td>
<td>See undefined_p.</td>
</tr>
<tr>
<td>undefined symbol</td>
<td>undefined_symbol_p</td>
<td>See undefined_symbol_p.</td>
</tr>
</tbody>
</table>

**Predicates**

Predicates are used to test a Gamma object for a given type, as listed. If a Gamma object is of that type, the predicate will return the value `t`.

**Syntax**

```
alist_p (s_exp)
array_p (s_exp)
autotrace_p (s_exp)
breakpoint_p (s_exp)
buffer_p (s_exp)
builtin_p (s_exp)
class_p (s_exp)
cons_p (s_exp)
constant_p (s_exp)
destroyed_p (s_exp)
file_p (s_exp)
fixed_point_p (s_exp)
function_p (s_exp)
instantiate_p (s_exp)
int_p (s_exp)
list_p (s_exp)
long_p (s_exp)
macro_p (s_exp)
method_p (s_exp)
nil_p (s_exp)
```
number_p (s_exp)
real_p (s_exp)
registered_p (s_exp)
string_p (s_exp)
sym_alist_p (s_exp)
symbol_p (s_exp)
true_p (s_exp)
none_p (s_exp)
undefined_p (s_exp)
undefined_symbol_p (s_exp)

Arguments

any expression

Returns

t or nil.

Example

Here is an example for the predicate function_p. All the other predicates work in a similar way.

Gamma> function_p(div);
t
Gamma> function_p(strcmp);
t
Gamma> function_p(5);
nil
Gamma>
undefined_p, undefined_symbol_p

undefined_p, undefined_symbol_p — test for undefined types and symbols.

Synopsis

<table>
<thead>
<tr>
<th>undefined_p (s_exp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>undefined_symbol_p (s_exp)</td>
</tr>
</tbody>
</table>

Arguments

s_exp
Any Gamma or Lisp expression.

Returns

t if the value of s_exp is not defined; otherwise nil.

Description

These two functions perform a similar task, checking to see if the s_exp is defined. However, they differ in two important ways:

• undefined_p examines the value of s_exp directly, whereas undefined_symbol_p expects the value of s_exp to be a symbol, and examines the value of that resulting symbol.

• undefined_p evaluates its argument in a protected scope where any "Symbol is undefined" errors will be trapped and disregarded. undefined_symbol_p evaluates its argument without protection, so it is possible that a "Symbol is undefined" error could be thrown if the evaluation of s_exp generates such an error.

Example

```
Gamma> a = #xyz
xyz
Gamma> undefined_p (a);
nil
Gamma> undefined_symbol_p (a);
t
Gamma> xyz = t;
t
Gamma> undefined_symbol_p (a);
nil
```
Gamma> undefined_p (y);
  t
Gamma> undefined_symbol_p (y);
Symbol is undefined: y
deqab 1>

See Also

Data Types and Predicates
Literals

— defined for integers, reals, strings, and symbols.

Integers

An integer is any group of digits defining a number between -2e+31 and 2e+31 - 1. It cannot contain a decimal point or an exponent. Integers have several different literal notations, but regardless of notation, all integers are 32 bit signed numbers. They are flagged internally with their respective notations and Gamma attempts to maintain and return the notation when the integer is printed.

Table 4. Integers

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal notation</td>
<td>539</td>
</tr>
<tr>
<td>0b</td>
<td>Binary notation</td>
<td>0b1011</td>
</tr>
<tr>
<td>0o</td>
<td>Octal notation</td>
<td>0o462</td>
</tr>
<tr>
<td>0x</td>
<td>Hexadecimal notation</td>
<td>0x35fc</td>
</tr>
<tr>
<td>'</td>
<td>Contents are a character</td>
<td>'M'</td>
</tr>
</tbody>
</table>

Real numbers

A real number is any group of digits defining a number less than -2e+31, greater than 2e +31 - 1, or containing a non-zero mantissa. It can contain a decimal point, and it may end with the letter e followed by a signed exponent.

Table 5. Real numbers

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0-9],[0-9]e[+</td>
<td>-][0-9]</td>
<td>Double-precision 64 bit floating-point number.</td>
</tr>
</tbody>
</table>

There are four pre-defined constants in Gamma:

PI      The value of pi, approximately 3.14159.
E       The base of the natural logarithm, approximately 2.71828.
INF     Floating point positive infinity.
NAN     Floating point not-a-number.

NAN is a floating point number that represents an invalid state. For example, 1 / x is a mathematical function that should produce a numeric result. If x is 0 then the numeric result will be not a number, but it will still be represented in floating point for the purpose of storage and future math functions.

For example:
Math on `NAN` is legal. It just produces another `NAN`. Math on strings is illegal. But, in the case of `1/0`, we catch that and throw an error. Mathematically the result may be `NAN` or `INF`, but in Gamma division by zero is an error. So you cannot use `1/0` to produce a `NAN` or `INF`.

**Strings**

A string may have any number of characters. The special forms \n, \t, \f and \r denote newline, tab, form feed, and carriage return respectively. The double quote (`"`) and back-slash (`\`) characters may be embedded in a string by preceding them with a backslash.

<table>
<thead>
<tr>
<th>Table 6. Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notation</strong></td>
</tr>
<tr>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

**Symbols**

Generally, symbol names are made up of alpha-numeric characters and underscores.

<table>
<thead>
<tr>
<th>Table 7. Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notation</strong></td>
</tr>
<tr>
<td>[a-z,A-Z,0-9]</td>
</tr>
<tr>
<td>_</td>
</tr>
<tr>
<td>\</td>
</tr>
</tbody>
</table>
Literals

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>be preceded by a back-slash to be used in a symbol name.</td>
<td></td>
</tr>
</tbody>
</table>

Other Data Types

The literal representation for all other Gamma data types is discussed in the reference entry associated with creating or accessing that data type, as given in the table below.

Table 8. Other Data Types

<table>
<thead>
<tr>
<th>Data type</th>
<th>Reference entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>array</td>
</tr>
<tr>
<td>Buffer</td>
<td>buffer</td>
</tr>
<tr>
<td>List</td>
<td>list</td>
</tr>
<tr>
<td>Instance</td>
<td>new</td>
</tr>
<tr>
<td>Function</td>
<td>function</td>
</tr>
<tr>
<td>Method</td>
<td>method</td>
</tr>
<tr>
<td>Class</td>
<td>class</td>
</tr>
<tr>
<td>File</td>
<td>open</td>
</tr>
<tr>
<td>Task</td>
<td>locate_task</td>
</tr>
</tbody>
</table>
Predefined Symbols

— a table.

Table 9. Symbols that are predefined in Gamma

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>Description</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>all_tasks</em></td>
<td>The list of tasks opened using locate_task.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>auto_load-alist</em></td>
<td>A list of rules used by AutoLoad.</td>
<td>read/write</td>
</tr>
<tr>
<td><em>case_sensitive</em></td>
<td>Used by reader to control case sensitivity. If nil, then all symbols are treated as lower-case. Default is t.</td>
<td>read/write</td>
</tr>
<tr>
<td><em>comma</em></td>
<td>Internal symbol.</td>
<td>not available</td>
</tr>
<tr>
<td><em>commasplice</em></td>
<td>Internal symbol.</td>
<td>not available</td>
</tr>
<tr>
<td><em>current_input</em></td>
<td>The currently open file for reading.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>debug</em></td>
<td>Not used.</td>
<td>not available</td>
</tr>
<tr>
<td><em>eof</em></td>
<td>Gamma representation of the end-of-file status from a read operation.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>eol</em></td>
<td>Gamma representation of the end-of-line status from a read operation.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>error_stack</em></td>
<td>The stack at the time the last error occurred.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>eval_silently</em></td>
<td>If set to t, then references to undefined symbols are returned as <em>undefined</em> instead of stopping the program with an error.</td>
<td>read/write</td>
</tr>
<tr>
<td><em>eval_stack</em></td>
<td>Contains the definition of the function being currently evaluated.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>event</em></td>
<td>The QNX Windows event name.</td>
<td>read-only</td>
</tr>
<tr>
<td>Symbol Name</td>
<td>Description</td>
<td>Accessibility</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td><em>fixed_point</em></td>
<td>Controls whether calculations with reals are done in double or fixed-point.</td>
<td>read/write</td>
</tr>
<tr>
<td><em>gui</em></td>
<td>The name of the graphical user interface that this version of Gamma was compiled against, as a string.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>gui_version</em></td>
<td>The version number of the graphical user interface that this version of Gamma was compiled against, as a string.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>ipc_file</em></td>
<td>String file used by IPC functions to create buffers for send/receive/reply sequence.</td>
<td>not available</td>
</tr>
<tr>
<td><em>jump_stack</em></td>
<td>Internal symbol.</td>
<td>not available</td>
</tr>
<tr>
<td><em>last_error</em></td>
<td>String containing last error.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>load_extensions</em></td>
<td>List of strings containing shell-match patterns of acceptable input files.</td>
<td>read/write</td>
</tr>
<tr>
<td><em>os</em></td>
<td>The name of the operating system (OS) that this version of Gamma was compiled in, as a string.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>os_version</em></td>
<td>The version number of the operating system that this version of Gamma was compiled in, as a string.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>os_release</em></td>
<td>The release number of the operating system that this version of Gamma was compiled in, as a string.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>require_path</em></td>
<td>List of strings of paths to search for require and require_lisp.</td>
<td>read-write</td>
</tr>
<tr>
<td><em>signal_handlers</em></td>
<td>See signal.</td>
<td>read-only</td>
</tr>
<tr>
<td>SIGABRT</td>
<td>See signal.</td>
<td>read-only</td>
</tr>
<tr>
<td>SIGALRM</td>
<td>See signal.</td>
<td>read-only</td>
</tr>
<tr>
<td>SIGBUS</td>
<td>See signal.</td>
<td>read-only</td>
</tr>
<tr>
<td>Symbol Name</td>
<td>Description</td>
<td>Accessibility</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SIGCHLD</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGCONT</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGFPE</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGHUP</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGILL</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGINT</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGIO</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGIOT</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGKILL</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGPIPE</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGPOLL</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGPWR</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGQUIT</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGSEGV</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGSTOP</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGTERM</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGTRAP</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGTSTP</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGTTIN</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGTTOU</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGURG</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGUSR1</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGUSR2</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
<tr>
<td>SIGWINCH</td>
<td>See <code>signal</code></td>
<td>read-only</td>
</tr>
</tbody>
</table>

__timers__

An array of active timers, in this format:

```
[[secs nsecs fires ((s-exp ...))...] number]...
```

- **secs**
  - The clock time in seconds when the timer was set.

- **nsecs**
  - The additional nanoseconds when the timer was set.
<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>Description</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>set.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fires</td>
<td>The set interval of time to fire, in seconds.</td>
<td></td>
</tr>
<tr>
<td>s-exp</td>
<td>Action(s) associated with the timer, inside a list of lists.</td>
<td></td>
</tr>
<tr>
<td>number</td>
<td>The timer number.</td>
<td></td>
</tr>
<tr>
<td><em>undefined</em></td>
<td>The Gamma representation of the undefined symbol state.</td>
<td>read-only</td>
</tr>
<tr>
<td><em>unwind_stack</em></td>
<td>The stack at the time that an error was recovered.</td>
<td>read-only</td>
</tr>
<tr>
<td>&amp;noeval , !</td>
<td>Symbol directing Gamma to not evaluate the next argument.</td>
<td>not available</td>
</tr>
<tr>
<td>&amp;optional , ?</td>
<td>Symbol directing Gamma to treat the following argument as optional.</td>
<td>not available</td>
</tr>
<tr>
<td>=&gt;&amp;rest , ...</td>
<td>Symbol directing Gamma to expect an optional number of arguments starting at last argument. Passed as a list.</td>
<td>not available</td>
</tr>
</tbody>
</table>
Reserved Words

— a table.

The following table provides a list of words which are reserved by the Gamma language. No symbols can be defined by the user that are identical to these reserved words.

**Table 10. Words reserved in Gamma**

<table>
<thead>
<tr>
<th>Reserved Word</th>
<th>Used In</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>class</code></td>
<td>Class declaration</td>
</tr>
<tr>
<td><code>collect</code></td>
<td><code>with</code> loop</td>
</tr>
<tr>
<td><code>do</code></td>
<td><code>with</code> loop</td>
</tr>
<tr>
<td><code>else</code></td>
<td><code>if</code> statement</td>
</tr>
<tr>
<td><code>for</code></td>
<td><code>for</code> loop</td>
</tr>
<tr>
<td><code>function</code></td>
<td>Function declaration</td>
</tr>
<tr>
<td><code>if</code></td>
<td><code>if</code> statement</td>
</tr>
<tr>
<td><code>local</code></td>
<td>Local variable declaration</td>
</tr>
<tr>
<td><code>method</code></td>
<td>Method declaration</td>
</tr>
<tr>
<td><code>tcollect</code></td>
<td><code>with</code> loop</td>
</tr>
<tr>
<td><code>while</code></td>
<td><code>while</code> loop</td>
</tr>
<tr>
<td><code>with</code></td>
<td><code>with</code> loop</td>
</tr>
</tbody>
</table>
Synopsis

The special object, $t$, is a logically true value which has no other meaning. All Gamma objects other than $\text{nil}$ are logically true, but only the special object $t$ is the logical negation of $\text{nil}$. $t$ is created by a call to not($\text{nil}$), or by reading the symbol $t$.

The predicate $\text{true\_p}$ explicitly tests for the value $t$. However, in all conditional statements, any non-$\text{nil}$ value is considered to be true for the purpose of the test.

Example

```
Gamma> x = 3;
3
Gamma> x > 2;
t
Gamma> x == 3;
t
Gamma> !nil;
t
Gamma> 10 < 25;
t
Gamma>
```

See Also

$, \text{nil}$
nil

nil — the logically false value.

Synopsis

| nil |

Returns

| nil |

Description

The special value, nil, is a zero-length list. It is the only logically false value in Gamma. All other Gamma values are considered to be logically true. A common mistake for first-time Gamma programmers is to treat the number zero as logically false.

Example

Gamma> x = 5;
5
Gamma> x > 10;
nil
Gamma> int_p(x);
t
Gamma> real_p(x);
nil
Gamma> !3;
nil
Gamma> !t;
nil
Gamma>
gamma, phgamma

gamma, phgamma — start Gamma and Gamma/Photon from the shell prompt.

Synopsis

gamma [-options] [program_name [program_arg]...]  
phgamma [-options] [program_name [program_arg]...]

Options

- c  command
  Execute the named command.

- C
  Declare all constants at startup.

- d
  Keep file and line # information on all cells.

- e
  Do not enter interactive mode.

- f filename
  'Require' (read and process) the named file and set the –e flag. As many files as desired can be processed by repeating this option. Although the file is run just like the executable named in program_name, the two are not the same, because no program arguments can be passed to a file using the –f option. When the file has been completely processed, Gamma moves on to the next option, if any, and will not necessarily enter the interactive mode.

- F
  Declare all functions at startup.

- G
  Run as Gamma, regardless of name.

- h
  Print a help message and exit.

- H heapsize
  Set the heap growth rate increment (default 2000).

- i filename
  'Require' the named file. This is identical to the –f option, except that Gamma will enter the interactive mode after all options have been processed.

- I
  Force entry into interactive mode after completion of the named application.
-L
  Run as Lisp, regardless of name.

-m
  Do not run the main function automatically.

-p
  Protect functions from the garbage collector. (Functions should not be redefined.)

-q
  Do not print copyright notice.

-s
  Set the local stack size in longwords.

-V
  Print the version number.

-X
  Exit immediately (usually used with -V).

_program_name
  The name of an executable program.

_program_arg
  The program arguments.

Returns

A Gamma prompt.

Description

This command starts Gamma or Gamma/Photon in interactive mode at the shell prompt. Flags are processed in the order given on the command line, and can appear more than once.

If the name of the executable contains the word 'Lisp', then it will use the Lisp grammar, otherwise it will use the Gamma grammar.

The -c and -f used together make possible several interesting ways to invoke and use Gamma. For example:

```
gamma -f domainA.g -c "init = methodA(3);" my_application "thing"
```

permits a user to specify a particular file to be processed, perhaps containing application-specific methods, then execute an arbitrary initialization expression, and finally start the intended application with specified arguments.

The -c argument used with -e has Gamma execute a command and exit without going into interactive mode. For example:
would load the Tower of Hanoi code, print the solution to the 3-disk hanoi problem, and then exit. (The single quotes are used to hide the double quotes from the shell.)

Example

```
[~/usr/devtools]$ gamma -m
Gamma(TM) Advanced Programming Language
Copyright (C) Cogent Real-Time Systems Inc., 1996-2001. All rights reserved.
Version 4.0 Build 31 at Aug 12 2001 09:57:56
Gamma>
```
Operators
Operator Precedence and Associativity

— a table.

Table 11. Operator Precedence and Associativity

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operator</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowest</td>
<td>ELSE</td>
<td>Right</td>
</tr>
<tr>
<td></td>
<td>=</td>
<td>Right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&amp; &amp;</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>&lt;, &gt;, &lt; =, &gt; =, = =, !=</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&amp;</td>
</tr>
<tr>
<td></td>
<td>-, +</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>Unary -, +, !</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>^</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>+, -, -=</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>[ ]</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>\ ..</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>()</td>
<td>Left</td>
</tr>
<tr>
<td>Highest</td>
<td>#</td>
<td>Left</td>
</tr>
</tbody>
</table>

The associativity of operators refers to order in which repeated use of the same operator will be evaluated. For example, the expression 1+2+3 will be evaluated as (1+2)+3 since the "+" operator associates the leftmost operator instances first. In contrast, the statement A = B = C will first perform the B = C assignment, and then the result is assigned to A.

Associativity should not be confused with precedence, which determines which one of different operators will be evaluated first. In the example 1+2 __3+4, the multiplication is performed first due to precedence, while the left addition is performed before the rightmost addition due to associativity, causing the expression to be evaluated as (1+ (2 __3) ) +4.

See Also

Arithmetic Operators, Assignment Operators, Bitwise Operators, Comparison Operators, Increment and Decrement Operators, Logical Operators, Quote Operators
Arithmetic Operators

Arithmetic Operators — (+, −, *, /, %)

Synopsis

\[
\begin{align*}
\text{number} + \text{number} \\
\text{number} - \text{number} \\
\text{number} \times \text{number} \\
\text{number} / \text{number} \\
\text{number} \% \text{number}
\end{align*}
\]

Arguments

\text{number}

Any integer or real number. Non-numbers are treated as zero.

Returns

The mathematical result of the operation.

Description

These operators perform simple mathematical operations on their arguments.

+ gives the sum of the two arguments.

− gives the difference between the first and second arguments.

∗ gives the product of the two arguments.

/ gives the first argument divided by the second argument.

% gives the modulus of the first argument by the second, that is, the remainder of the integer division of the first argument by the second.

Example

\[
\begin{align*}
\text{Gamma}> & \ 5 + 6; \\
& 11 \\
\text{Gamma}> & \ 12 / 5; \\
& 2.3999999999999999112 \\
\text{Gamma}> & \ \text{div}(12, 5); \\
& 2 \\
\text{Gamma}> & \ 19 \% 5; \\
& 4
\end{align*}
\]
Gamma >
Assignment Operators

Assignment Operators — (=, :=, :::)

Synopsis

\[
\begin{align*}
symbol &= s_{\exp} \\
symbol &:= s_{\exp} \\
symbol &::= s_{\exp}
\end{align*}
\]

Arguments

- **symbol**
  - Any valid symbol.
- **s_exp**
  - Any expression.

Returns

The assigned value.

Description

- `=` is used to assign a value to a variable.
- `:=` is used to assign a value only if no value is currently assigned to the `symbol`. If the `symbol` already has a value then the `symbol` keeps its original value.
- `::=` is used to assign a constant. Once the assignment has been made no changes to the `symbol` are allowed. Attempted changes to the `symbol` will generate an error.

Example

\[
\begin{align*}
\text{Gamma}> & \ a = 5; \\
& 5 \\
\text{Gamma}> & \ a := 6; \\
& 5 \\
\text{Gamma}> & \ a; \\
& 5 \\
\text{Gamma}> & \ b ::= 7; \\
& 7 \\
\text{Gamma}> & \ b = 8; \\
\text{Assignment to constant symbol: b} \\
\text{debug 1>}
\end{align*}
\]
Gamma> b ::= 9;
Defvar of defined constant: b
debug 1>
Gamma>
Binary Operator Shorthands

Binary Operator Shorthands — (+=, -=, *=, /=, %=, &=, ^=, <<=, >>=)

Synopsis

```
symbol += number
symbol -= number
symbol *= number
symbol /= number
symbol %= number
symbol &= number
symbol ^= number
symbol <<= number
symbol >>= number
```

Arguments

- **symbol**: A symbol with a numeric value.
- **number**: Any integer or real number.

Returns

The value of the **symbol** as operated on with the **number**.

Description

These operators provide a shorthand way of reassigning values to symbols.

- `+=` gives the sum of the **symbol** and the **number**.
- `-=` gives the difference between the **symbol** and the **number**.
- `*=` gives the product of the **symbol** and the **number**.
- `/=` gives the **symbol** divided by the **number**.
- `%` gives the modulus of the **symbol** by the **number**, that is, the remainder of the integer division of the **symbol** by the **number**.
- `&=` performs the `&` operation on the **symbol** and the **number**.
- `^=` performs the `^` operation on the **symbol** and the **number**.
- `<<=` performs the `<<` operation on the **symbol** and the **number**.
Binary Operator Shorthands

$\gg=\ $ performs the $\gg$ operation on the symbol and the number.

Example

```
Gamma> a = 5;
5
Gamma> a += 8;
13
Gamma> a;
13
Gamma>
```

See Also

Arithmetic Operators, Assignment Operators, Bitwise Operators
Bitwise Operators

Bitwise Operators — (<<, >>, ~, &, |, ^)

Synopsis

number << shift;
number >> shift;
~ number
number & number
number | number
number ^ number

Arguments

number
Any number,
shift
The number of bit shifts to perform.

Returns

An integer which is the result of the particular operation.

Description

<<, >> return the first argument with a left or right bitshift operation performed the number of times of the second argument.

~ returns the binary opposite of the number.

& compares each of the corresponding digits of the two numbers. If both digits are 1, returns 1 for that place. Otherwise returns 0 for that place.

| compares each of the corresponding digits of the two numbers. If either those digits is 1, returns 1 for that place. Otherwise returns 0 for that place.

^ compares each of the corresponding digits of the two numbers. If both digits are the same, returns 0 for that place. If they are different (ie. 0 and 1) returns 1 for that place.

Examples

Gamma> bin(10);
0b1010
Gamma> bin(10 << 1);
0b00010100
Gamma> bin(10 >> 1);
0b0101
Gamma> bin (~10);
0b111111111111111111111111110101
Gamma> bin(10);
0b1010
Gamma> bin (9);
0b1001
Gamma> bin (9 & 10);
0b1000
Gamma> bin (9 | 10);
0b1011
Gamma> bin (9 ^ 10);
0b0011
Gamma>

See Also

band, bnot, bor, bxor
Class Operators

Class Operators — (., .)

Synopsis

```
instance.variable = value
instance.variable
instance..variable = value
instance..variable
```

Arguments

- **instance**
  - An instance of a class.
- **variable**
  - An instance variable name.
- **value**
  - A new value to write to the instance variable.

Returns

The value argument.

Description

These operators assign and evaluate object instance values, using familiar C/C++ structure/class reference syntax. The *instance* and its instance *variable* are separated by a period and the assignment is made using the *assignment operator*. Using two periods between *instance* and *variable* makes the reader interpret the instance variable.

Using either the . or the .. without the = assignment operator causes the *variable* to be evaluated at that instance.

Examples

```
Gamma> class cmpny { name; address; }
(defclass cmpny nil [])[address name])
Gamma> company = new(cmpny);
{cmpny (address) (name)}
Gamma> company.name = "Acme Widgets";
"Acme Widgets"
Gamma> company.name;
"Acme Widgets"
```
Here is an example of how the .. syntax can be used to allow an instance of one class to access a method of another class. This can be useful if a parent and child widget have different methods with the same name, and you want an instance of one to use the method of the other.

Gamma> var = symbol("name");
name
Gamma> company..var;
"Acme Widgets"
Gamma>

Gamma> class A{}
(defclass A nil [[]])
Gamma> class B{}
(defclass B nil [[]])
Gamma> class C B{}
(defclass C B [[]])
Gamma> method A.get (){princ("Class A's method.
");}
(defun A.get (self) (princ "Class A's method.
"))
Gamma> method B.get (){princ("Class B's method.
");}
(defun B.get (self) (princ "Class B's method.
"))
Gamma> a = new(A);
{A}
Gamma> a.get();
Class A's method.
Gamma> b = new(B);
{B}
Gamma> b.get();
Class B's method.
Gamma> (b..A.get());
Class A's method.
Gamma> (a..B.get());
Class B's method.
Gamma> c = new(C);
{C}
Gamma> (c..A.get());
Class A's method.
Gamma> (c..B.get());
Class B's method.
Comparison Operators

Comparison Operators — (!=, <, <=, ==, >, >=)

Synopsis

```
number != number
number &lt; number
number &lt;= number
number == number
number &gt; number
number &gt;= number
```

Arguments

`number`

Any integer or real number. Non-numbers are treated as zero.

Returns

- `!=`  
  - `t` if the first `number` is not equal to the second, else `nil`.
- `<`  
  - `t` if the first `number` is less than the second, else `nil`.
- `<=`  
  - `t` if the first `number` is less than or equal to the second, else `nil`.
- `==`  
  - `t` if the first `number` is equal to the second, else `nil`.
- `>`  
  - `t` if the first `number` is greater than the second, else `nil`.
- `>=`  
  - `t` if the first `number` is greater than or equal to the second, else `nil`.

Description

These functions perform a numeric comparison of their arguments. In mathematical (infix) notation, the function would put the first argument on the left side of the comparison, and the second argument on the right side of the comparison.

Example

```
Gamma> 5 < 6;
t
Gamma> 5 > 6;
nil
Gamma> 5.00 == 5;
t
```

Comparison Operators

Gamma> "hello" == string("hel","lo");
t
Gamma> a = 5 + 1;
6
Gamma> a;
6
Gamma> a == 5;
nil
Gamma>

See Also

eq, equal, strcmp, stricmp
Evaluation Order Operators

Evaluation Order Operators — , ( )

Synopsis

\[ \text{symbol, symbol} \]
\[ ( \text{symbol operator symbol} ) \]

Arguments

\text{symbol}

- Any symbol.

\text{operator}

- Any operator.

Determine

Sequence of operation.

Description

Operations before a , are performed before those after it.

Operations enclosed by ( and ) are performed first.

Examples

\begin{verbatim}
Gamma> x = 3;
x = 3
Gamma> princ("x = ", x, "\n");
x = 3
Gamma> (2 + 3) * 4;
14
Gamma> 2 + (3 * 4);
14
Gamma>
\end{verbatim}
**Increment and Decrement Operators**

Increment and decrement operators — (++, --)

**Synopsis**

```
++symbol
symbol++
--symbol
symbol--
```

**Arguments**

`symbol`

A symbol whose value is a number.

**Returns**

The value of the symbol plus or minus one.

**Description**

These operators perform auto-increments or decrements on numeric symbols. When the `++` is placed before a symbol, it performs a pre-increment, where the value is incremented and the result is the symbol's value + 1. When `++` is placed after a symbol, it performs a post-increment. Here the result of the operation is the value of the symbol prior to being incremented. The `--` operator works in the same way. These operators only take symbols as arguments. It is not possible to auto-increment or decrement an array element, list element, or instance variable.

**Examples**

```
Gamma> a = 5;
5
Gamma> ++ a;
6
Gamma> a;
6
Gamma> a ++;
6
Gamma> a;
7
Gamma> a = 5;
5
Gamma> -- a;
4
```
4
Gamma> a;
4
Gamma> a --;
4
Gamma> a;
3
Gamma>
Logical Operators

Logical Operators — (!, &&, ||)

Synopsis

! s_exp  
s_exp && s_exp ...  
s_exp || !s_exp ...

Arguments

s_exp  
Any Gamma or Lisp expression.

Returns

Non-nil or nil.

Description

In Gamma or Lisp, any expression which is not nil is treated as being true (t) for the purpose of boolean logic. Applying ! to any non-nil expression will produce nil. Applying ! to nil must produce an arbitrary non-nil result. The generic non-nil value in Gamma is t.

&& evaluates each of its arguments in order, and continues so long as each argument evaluates to non-nil. If any argument is nil, then nil is returned immediately, without evaluating the rest of the arguments. If no argument is nil, the last argument is returned.

|| returns non-nil if any of its arguments is not nil. Each argument is evaluated in turn, and as soon as a non-nil value is reached, that argument is returned. Subsequent arguments are not evaluated.

Examples

Gamma> 6;  
6  
Gamma> !6;  
nil  
Gamma> !nil;  
t  
Gamma> 5<6 && string("hi ","there");  
"hi there"  
Gamma> 5>6 && string("hi ","there");  
nil
Logical Operators

\begin{verbatim}
Gamma> x = 5;
5
Gamma> y = 6;
6
Gamma> (x = t) || (y = 0);
t
Gamma> x;
t
Gamma> y;
6
Gamma>
\end{verbatim}

See Also

and, not, or
Quote Operators

Quote Operators — (#, `, @)

Synopsis

```
# s_exp
` s_exp
@ s_exp
```

Arguments

`s_exp`

Any Gamma or Lisp expression.

Returns

Does not evaluate the symbol; it returns the protected expression that follows it.

Description

Normally Gamma evaluates every expression as it parses through the code. The `#` operator protects the contents of an expression from the evaluator. The `.` operator does the same thing, but allows for evaluation of sub-expressions. Any sub-expression tagged with `@` operator that occurs within a back-ticked (``) expression will be evaluated.

Any error messages involving the `@` operator will use the expression _comma_. This is because the `@` operator in Gamma corresponds to a comma operator (,) in Lisp syntax. When a Gamma expression is passed to Lisp, the `@` operator is converted to a comma. But if the Lisp comma operator is ever read back into Gamma, it is represented by the symbol _comma_ to avoid confusion with the (,) operator used in Gamma function calls.

Examples

```
Gamma> name = "John";
"John"
Gamma> name;
"John"
Gamma> #name;
name

Gamma> x = 4;
4
Gamma> list (1,x);
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
Gamma> #list (1,x);
(list 1 x)
Gamma> list (1,#x);
(1 x)
Gamma> `list (1,x);
(list 1 x)
Gamma> `list (1,@x);
(list 1 4)
Symbol Character Operators

Symbol Character Operators — (\, $)

Synopsis

\symbol_character
$symbol_character_string

Arguments

symbol_character
A character that is normally not valid within the string of a symbol name.

symbol_character_string
A symbol name that contains one or more characters that are normally not valid within the string of a symbol name.

Returns

A valid symbol name.

Description

These operators allow you to put non-valid characters into a symbol name. They must be used every time the symbol is written, not just the first time.

\ makes the immediately following character valid.

$ makes the whole string valid, regardless of which individual characters are not normally valid.

Example

Gamma> my\:example1 = 5;
5
Gamma> x = my\:example1 + 7;
10
Gamma> $my:example2 = 9;
9
Ternary Operator

Ternary Operator — ( ? : )

Synopsis

\[ condition \ ? \ s\_exp \ : \ s\_exp \]

Arguments

- **condition**
  - Any Gamma or Lisp expression.
- **s\_exp**
  - Any Gamma or Lisp expression.

Returns

The first \textit{s\_exp} if the \textit{condition} is true, otherwise the second \textit{s\_exp}.

Examples

```
Gamma> a = t ? 2 : 8;
2
Gamma> a;
2
Gamma> b = (a == 7) ? 2 : 8;
8
Gamma> b;
8
Gamma>
```
Statements
class

class — defines a class.

Synopsis

class class_name [parent]
{
    [instance_var [= initial_value];]
    ...
    [static: class_var [= initial_value];]
    ...
}

Arguments

class_name
    The name of the new class.

parent
    The parent (base) class.

class_var
    Class variable definition.

instance_var
    Instance variable definition. This is provided in the form of a list of variable definitions. Each variable definition is either a variable name or a list which contains a variable name and a default value expression. Whenever a new instance is formed, the default value expression is evaluated to the default value. If no default value is given, the instance variable's value will be nil.

initial_value
    Initial value given to instance_var, if none then nil is assigned to that instance variable.

Returns

    A class definition.

Description

    This function constructs a class definition and binds the class-name symbol in the current scope to refer to that class. The class mechanism allows only a single parent (base) class.
None of the arguments to `class` is evaluated. If `instance_vars` are defined with the same names as inherited variables, the inherited variables are overridden and cannot be accessed by instances of this class.

- The `class` statement creates a new class.
- If the parent (base) class is omitted, or is `nil`, then the resulting class has no parent (base).
- Each instance variable consists of a name (a symbol) and an optional initial value that will be assigned whenever a new instance of the class is created using the `new` function.
- The resulting class definition, which is a data object in its own right, will be assigned to the symbol, `name`.

**Example**

This example creates two classes: a base class, `RegPolygon`; and a class derived from it, `Square`. `RegPolygon` has two attributes: `sides` and `length`. When `Square` is created, its parent (base) class (`RegPolygon`) is explicitly assigned. In addition, the attribute `sides` is assigned a value of 4.

```
Gamma> class RegPolygon {sides; length;
(defclass RegPolygon nil [] [length sides])
Gamma> class Square RegPolygon {sides = 4;
(defclass Square RegPolygon [] [length (sides . 4)])
```

This example creates a class with instance variables and class variables.

```
Gamma> class Other {ivar1; ivar2; static: cvar1; cvar2;}
(defclass Other nil [cvar1 cvar2][ivar1 ivar2])
Gamma>
```

**See Also**

`Class Operators`, `class_add_cvar`, `class_add_ivar`, `method`, `new`
condition

condition — tests conditions.

Synopsis

condition {case condition: statement
    [case condition: statement]
    ...
    [default: statement]}

Arguments

condition
    Any Gamma or Lisp expression.
statement
    Any Gamma statement.

Returns

The return value of the statement that corresponds to the first true condition or the default. Otherwise nil.

Description

This statement is similar to the switch statement, except that it takes no arguments. It checks the truth value of each condition in turn. The first true condition encountered returns with the return value of the passed statement. If no condition is true, it returns the return value of the default statement, (or nil, if no default statement is given).

The words "case" and "default" and the symbols {, :, and } are unchanging syntactical elements.

Example

Gamma> a = 5;
5
Gamma> b = 9;
9
Gamma> condition {case a == b: princ("Equal\n");
    case a != b: princ("Unequal\n");}
Unequal
Gamma>
Also see the example in \texttt{switch}.

\textbf{See Also}

\texttt{switch}
for

for — checks a condition and performs a statement.

Synopsis

for (setup ; condition ; iteration) statement

Arguments

setup
An iteration setup, usually a variable with an initial value.

condition
The condition to test.

iteration
Any Gamma expression, usually used to increment the variable in setup.

statement
Any Gamma statement.

Returns

The value of the condition.

Description

This statement is essentially identical to a for loop in C, and the syntax is the same. It checks a condition iteratively, and executes a statement when the condition is true.

Example

This for loop counts from 0 to 10, printing out the value of i as it loops.

```gamma
for (i=0;i<=10;i++)
{
    princ("value of i: ", i, "\n");
}
```

See Also

Statements, while, with
function

function — creates a new function.

Synopsis

```
function name ([argument [,argument]... ]) statement
```

Arguments

- `name`: The name of the function.
- `argument`: A symbol that names the argument.
- `statement`: The body of the function.

Returns

A named function definition in Lisp syntax. When a function is called, the return value is the value of the last expression to be evaluated in the function body.

Description

The `function` statement declares a new function. All function arguments are implicitly local to the scope of the function. The argument list is denoted by parentheses, and contains zero or more argument definitions, separated by commas. Each `argument` can be a symbol, which is the name of the argument, along with any combination of the following modifiers:

- `!` before the `argument` indicates that this argument will not be evaluated when the function is called.

- `?` after the `argument` indicates that this argument is optional. Only the first optional argument has to be marked as optional. All arguments after that are implicitly optional. An optional argument may have a default value, specified by appending `=` expression after the question mark.

  The only way to test whether an optional function parameter has been provided is by using the predicate `undefined_p`, which tests for the `_undefined_` value.

- `...` after the `argument` indicates that this argument is a "catch-all" argument used to implement variable length argument lists. Only the last argument in the argument list can
have ... after it. An argument modified by ... will always be either \texttt{nil}, or a list containing all arguments from this position onward in the function call.

When a function is called, its arguments are bound in a new local scope, overriding previous definitions of those symbols for the duration of the function.

\section*{Example}

This function, with one argument, returns an integer at least one greater than the argument.

\begin{verbatim}
function next (n)
{
    ceil(n) + 1;
}
\end{verbatim}

This function prints its first two arguments and optionally prints a new line (which is printed by default). It returns a string concatenation of the first two arguments.

\begin{verbatim}
function print_two (first, second, newline?=t)
{
    princ(first, " ", second);
    if (newline)
        terpri();
    string(first, " ", second);
}
\end{verbatim}

This function adds all of its arguments. It insists on having at least one argument. Notice that the optional character '?' and the rest character '...' are combined in the second argument.

\begin{verbatim}
function add_all (first, others...?)
{
    local sum,x = 0;
    sum = first;
    if ( !undefined_p(others) )
    {
        for(x=others;x=x=cdr(x))
            sum = sum + car(x);
    }
    sum;
}
\end{verbatim}

\section*{See Also}

\texttt{method}, \texttt{Statements}
if

if — conditionally evaluates statements.

Synopsis

```lisp
if (condition) statement [else statement]
```

Arguments

- `condition`:
  A Gamma or Lisp expression to test.

- `statement`:
  A statement to perform if the condition is non-nil.

Returns

The evaluation of the `statement` that corresponds to the satisfied `condition`.

Description

The `if` statement evaluates its `condition` and tests the result. If the result is non-nil, then the `statement` is evaluated and the result returned.

The `else` option allows for another statement. If the `condition` is nil, the `else` statement (if included) is evaluated and the result returned. This statement could be another `if` statement with another condition and `else` statement, etc., permitting multiple else/if constructs. The entire else option can be omitted if desired.

- If the `condition` is nil and no else `statement` exists, nil is returned.
- The else part of a nested if statement will always be associated with the closest if statement when ambiguity exists. Ambiguity can be avoided by explicitly defining code blocks (using curly brackets).
- In interactive mode Gamma has to read two tokens (`if` and `else`) before it will process the statement. If you are not using an `else` part, you have to enter a second semicolon (;) to indicate that the `if` statement is ready for processing.

Example

```lisp
Gamma> x = 5;
5
Gamma> y = 6;
6
```
The following code:

```c
name = "John";
age = 35;

if ((name == "Hank") || (name == "Sue"))
{
    princ("Hi ", name,"\n");
}
else if ((name == "John") && (age < 20))
{
    princ("Hi ", name," Junior\n");
}
else if ((name == "John") && (age >= 20))
{
    princ("Hi ", name," Senior\n");
}
else
{
    princ("I don't know you\n");
}
```

Will produce the following results:

```
Hi John Senior
```

See Also

Statements, for, while
local

local — allows for implementing local variables within functions.

Synopsis

    local !variable [= s_exp] [, !variable [= s_exp]...];

Arguments

    variable
        A symbol.

    s_exp
        Any Gamma or Lisp expression.

Returns

    The value of the s_exp, or nil if no value was assigned.

Description

This statement is provided to allow other grammars to implement local variables within functions. It defines new local variables in the current scope, overriding any outer scope that may also define those variables. Each local variable consists of a variable name (a symbol) and an optional initial value.

To test whether a symbol is bound in the current scope, use the predicate undefined_p.

Example

    Gamma> i = 5;
    5
    Gamma> function print_three_local ()
    {
        local i;
        for(i=1;i<=3;i++)
        {
            princ("value of i is: ", i, "\n");
        }
    }
    <function definition>
    Gamma> function print_three_global ()
    {
        // local i;
for(i=1;i<=3;i++)
{
    prin("value of i is: ", i, "\n");
}

This example shows the variable i receiving the value of 5. The two functions are defined identically, except for their names and where the second function comments out the 'local' command. When the first function is run the internal scoping using the local directive protects the value of i globally. When the function returns, i remains at 5, even though it was the value 1, 2 and 3 within the scope of that function.

The second function has the 'local' directive commented out (using //), so the global variable i is modified. When the function returns and we check the value of i, it is 4. The 'for-loop' within the second function incremented the value of i until it failed the i<=3 comparison. After the second function is run the value of i is 4. The global variable i has not been protected in the second function.
**method**

*method* — defines a method for a given class.

**Synopsis**

```
method class.method_name ([argument [, argument]...]) statement
```

**Arguments**

- **class**
  - The class for which this method is defined.
- **method_name**
  - The name of the method.
- **arguments**
  - The argument list for this method. This does not include the implied argument self, nor is self defined when the arguments are bound as the method is called.
- **statement**
  - The body code statement for this method. Within this statement the special variable `self` is defined as the instance on which this method is being applied.

**Returns**

A function definition of the resulting method function.

**Description**

This statement defines a method function for a given class. If a method already exists for this class with this name, the previous definition will be replaced. If a method of the same name exists for any parent (base) class of the given class, it will be overridden for instances of this class only. In Gamma methods are run using the syntax:

```
instance.method_name(arguments);
```

which is the familiar `object.method` syntax used in C++.

- The method syntax creates a new method for a particular class. It is an error to omit the class.
- The argument list for method is identical to the argument list for function.

**Example**

```
Gamma> class RegPolygon{sides; length;} (defclass RegPolygon nil [] [length sides])
```
Gamma> **method** RegPolygon.perimeter () { (* .sides * .length;)
(defun RegPolygon.perimeter (self) (* (self sides) (self length)))
Gamma> **class** Square RegPolygon { sides = 4; }
(defclass Square RegPolygon [] [length (sides . 4)])
Gamma> **method** Square.area () { sqr (self.length);)
(defun Square.area (self) (sqr (self length)))
Gamma> sqB = new (Square);
(Square (length) (sides . 4))
Gamma> sqB.length = 3;
3
Gamma> sqB.perimeter();
12
Gamma> sqB.area();
9
Gamma>

**See Also**

Class Operators, class, defun, new
progn, prog1

progn, prog1 — group several statements into one expression.

Synopsis

```
progn {!statement ![statement] ...}
prog1 {!statement ![statement] ...}
```

Arguments

`statement`

Any valid Gamma statement.

Returns

`progn`: the return value of the last statement.

`prog1`: the return value of the first statement.

Description

These two syntactical elements are not statements, but they transform a group of one or more statements into a single Gamma expression. The value of the resulting expression is the return value of the last statement for `progn` or the first statement for `prog1`. This is useful for performing complex actions where only a single expression is permitted, such as in a callback. No new scope is entered for a `progn` or a `prog1`.

The syntax of these unique elements uses curly braces `{ }`, but don’t confuse them with statements. They behave exactly like expressions.

Example

```
Gamma> a = 2;
2
Gamma> b = 5;
5
Gamma> progn(a = 3; princ("a: ",a,"\n"); c = a + b; princ("c: ",c,"\n"););
a: 3
c: 8
Gamma> string( prog1{a = 5; b = 1;} );
"5"
Gamma> a;
```
5
Gamma> b;
1
Gamma>
**protect unwind**

*protect unwind* — evaluates protected code, despite errors.

**Synopsis**

```
protect statement unwind statement
```

**Arguments**

*statement*

Any Gamma or Lisp statement.

**Returns**

If no error occurs, the result of evaluating the protect *statement* and the unwind *statement*. If an error occurs, the result of the unwind *statement* only.

**Description**

This function ensures that a piece of code will be evaluated, even if an error occurs within the protect *statement* code. This is typically used when an error might occur but cleanup code has to be evaluated even in the event of an error. The error condition will not be cleared by this statement. If an error occurs, control will be passed to the innermost *trap_error* function or to the outer level error handler immediately after the unwind *statement* is evaluated.

**Example**

This code will close its file and run a *write_all_output* function even if an error occurs.

```
if (fp=open("filename","w"))
{
    protect close(fp); unwind write_all_output();
}
```

**See Also**

*error, Statements, try catch, Tutorial II Error Handling*
**switch**

`switch` — tests arguments with conditions.

**Synopsis**

```
switch (symbol) {case condition:
    statement
    [statement...]

    [case condition:
        statement
        [statement...]]

    ...

    [default:
        statement
        [statement...]]}
```

**Arguments**

*symbol*

A symbol with a value to test against the value of the *condition*(s).

*condition*

Any Gamma or Lisp expression.

*statement*

Any Gamma statement.

**Returns**

The return value of the *statement* that corresponds to the first satisfied *condition* or the default. Otherwise `nil`.

**Description**

This statement is similar to the *condition* statement, except that it takes an argument. It checks the value of the passed *symbol* against the value of the *condition* for each case in turn. The first match returns the return value of the corresponding *statement*. If there is no match, it returns the return value of the default *statement*, if any.

The words "case" and "default" and the symbols {, :, and } are unchanging syntactical elements.
Example

Gamma> a = "on";
"on"
Gamma> b = 6;
6
Gamma> c = "Nothing";
"Nothing"
Gamma> switch(a) {case "on": 75; case "off": 20; default: 0;}
75
Gamma> switch(b) {case "on": 40; case "off": 10; default: princ("Huh?\n");
Huh?
Gamma> switch(c) {case "on": 1; case "off": 0;}
nil
Gamma>

#!/usr/cogent/bin/gamma

/*
   This example demonstrates the switch and condition statements. The switch statement checks the command line argument and prints a response. The case argument checks the command line argument and the result of the switch statement.
*/

function main ()
{
    a = number ((cadr(argv)));

    switch (a)
    {
        case 1:
            princ ("One\n");
        case 2:
            princ ("Two\n");
        case 2+1:
            princ ("Three\n");
        case 4:
            princ ("Four\n");
        default:
            princ ("Something else: ", a, "\n");
    }
condition
{
    case a == 1:
        princ ("Condition a == 1\n");
    case cadr(argv) == "Hello":
        princ ("Condition a == Hello\n");
    default:
        princ ("No condition met\n");
}
try catch

try catch — catches errors in the body code.

Synopsis

try statement catch statement

Arguments

statement
Any Gamma or Lisp statement.

Returns

If no error occurs, the result of evaluating the try statement. If an error occurs, the result of the catch statement.

Description

This statement catches any errors that may occur while evaluating the try statement code. If no error occurs, then try catch will finish without ever evaluating the catch statement. If an error does occur, try catch will evaluate the catch statement code immediately and the error condition will be cleared. This is usually used to protect a running program from a piece of unpredictable code, such as an event handler. If the error is not caught it will be propagated to the top-level error handler, causing the interpreter to go into an interactive debugging session.

Example

The following code:

#!/usr/cogent/bin/gamma

try
{
  2 + nil;
}
catch
{
  princ("Error:\n", _error_stack_, "\n");
}

Will give these results:

Error:
The following piece of code will run an event loop and protect against an unpredictable event.

```lisp
((trap_error #0=(+ 2 nil) (princ Error:
  _error_stack_
)) #0#)
```

See Also

- error
- Statements
- trap_error
- protect unwind
- Tutorial II Error Handling
while

while — iterates, evaluating a statement.

Synopsis

```
while (condition) statement
```

Arguments

- **condition**
  - Any Gamma or Lisp expression.

- **statement**
  - Any Gamma or Lisp statement.

Returns

The value of `condition` at the final iteration.

Description

This function iterates until its `condition` evaluates to `nil`, evaluating the `statement` at each iteration. The `condition` is evaluated before the `statement`, so it is possible for a `while` loop to iterate zero times.

Example

```
Gamma> x = 0;
0
Gamma> while (x < 5) { princ (x, "\n"); x++; }
0
1
2
3
4
Gamma> x;
5
Gamma>
```

See Also

* `for`, `if`, Statements*
with

with — traverses an array or list performing a statement.

Synopsis

```lisp
with symbol in|on s_exp do statement

with symbol1 in|on s_exp symbol2 = collect|tcollect statement
```

Arguments

- `symbol`
  - Any Gamma or Lisp symbol.
- `s_exp`
  - Any Gamma or Lisp expression.
- `statement`
  - Any statement.

Returns

- `nil` when using the `do` option, and the result of the `statement` when using the `collect` or `tcollect` option.

Description

```lisp
with symbol in|on s_exp do statement
```

- A `with` loop using the iteration style `in` traverses an array or list as defined by an expression (`s-exp`), performing the `statement` with the iteration `symbol` assigned to each element of the array or list in turn.
- A `with` loop using the iteration style `on` traverses a list defined by an expression (`s-exp`), performing the `statement` with the iteration `symbol` assigned to the car and then successive cdrs of the list.
- The iteration `symbol` is local in scope to the `with` statement.

```lisp
with symbol1 in|on s_exp symbol2 = collect|tcollect statement
```

- A `with` loop using the `collect` directive will collect the results of the `statement` for each iteration, and produce an array or list (depending on the type of the original array or list), whose elements correspond on a one-to-one basis with the elements of the original array or list.
- A `with` loop using the `tcollect` directive will collect the results of the `statement` at each iteration, ignoring `nil` results in the body. The resulting array or list will not have a
one-to-one correspondence with the original array or list.

- The result of a with loop using `collect` or `tcollect` will be assigned to `symbol2`, which is not local in scope to the `with` statement. The iteration `symbol1` is local in scope to the `with` statement.

**Examples**

```gamma
Gamma> A = array(1,2,3,4);
[1 2 3 4]
Gamma> with x in A do
{
    x = x + 1;
    princ(x, "\n");
}
2
3
4
5
nil

Gamma>

Gamma> L = list (1, 2, 3, 4, 5, 6);
(1 2 3 4 5 6)
Gamma> with x on L do (princ(x,"\n"));
(1 2 3 4 5 6)
(2 3 4 5 6)
(3 4 5 6)
(4 5 6)
(5 6)
(6)
nil
nil
Gamma> with x in L y = collect x + 2;
(3 4 5 6 7 8)
Gamma> y;
(3 4 5 6 7 8)
Gamma> with x in L y = tcollect x < 4 ? x : nil;
(1 2 3)
Gamma> y;
(1 2 3)
Gamma>
```

**See Also**

`for`, `if`, `Statements`
call

call — calls a class method for a given instance.

Synopsis

call (instance, method, !argument...)  
call (instance, class, method, !argument...)

Arguments

instance  
An instance of a class.

method  
A method name defined for the class of the instance.

class  
A class name.

argument  
The arguments to the method.

Returns

The result of calling the named method on the given instance with the provided arguments.

Description

This function explicitly calls a class method for the provided instance, using the same argument list as would be required for a call using (instance method ...) syntax in Lisp, or the instance.method (...) syntax in Gamma. The second syntax of this function provides a means for calling an explicit class method even if the class of the instance overloads the method name. Notice that the arguments to call are all evaluated.

Example

This example is based on the class and method developed in method.

Gamma> call(sqB, Square, #perimeter);
12
Gamma> call(sqB, Square, #area);
9
See Also

method
**class_add_cvar**

*class_add_cvar* — adds new class variables.

**Synopsis**

```
class_add_cvar (class variable init_value? type?);
```

**Arguments**

- **class**
  - The class receiving the new class variable.
- **variable**
  - The new variable to add to the class.
- **init_value**
  - Optional argument for initial value of the variable.
- **type**
  - Optional argument for the type of variable.

**Returns**

The value of the new argument.

**Description**

This function adds new class variables to either binary (built-in) or user-defined classes. Class variables are special variables that are available to be set/read by any instance of the class, as well as any derived classes or their instances, whether they were created before or after the class variable was defined.

**Example**

This example is based on the class and method developed in **method**.

```
Gamma> class_add_cvar(RegPolygon, #linethickness, 2);
2
Gamma> polyD = new(RegPolygon);
{RegPolygon (length) (sides)}
Gamma> polyD.linethickness;
2
Gamma> sqB.linethickness;
```
See Also

class_add_ivar
**class_add_ivar**

- adds an instance variable to a class.

**Synopsis**

```lisp
class_add_ivar (class, variable, init_value?, type?)
```

**Arguments**

- `class`: A class.
- `variable`: A symbol to be used as the instance variable name.
- `init_value`: Any Gamma or Lisp expression to be used as an initial value.
- `type`: An integer number which will be stored with the instance variable. This type is not used by the interpreter, and may be anything.

**Returns**

- The value of the `init_value`.

**Description**

This function dynamically adds an instance variable to a class. All instances of that class which are created after this call will contain this instance variable. All instances created before this call will not contain this instance variable. This function is typically called on a class before any instances are created. It is too late to call this function within an instance constructor. All subclasses of this class will inherit the new instance variable. If the ivar already exists on the class, the only effect of this function is to change the default value.

**Example**

This example is based on the class and method developed in `method`. The instance `sqB` does not have "color" as an instance variable because it was created before the instance variable "color" was added.

```
Gamma> class_add_ivar(Square, #color, "red", 12);
"red"
Gamma> sqC = new(Square);
{Square (color . "red") (length) (sides . 4)}
```
Gamma> sqB;
(Square (length . 3) (sides . 4))
Gamma>

See Also

instance_vars
class_name

class_name — gives the name of the class.

Synopsis

class_name (class|instance)

Arguments

class|instance

A class or instance of a class.

Returns

The name of the class, as a symbol.

Example

This example is based on the class and method developed in method.

Gamma> y = Square;
(deffunction Square (defun Square.area (self)
    (sqr (@ self length)))) [length (sides . 4)]
Gamma> class_name(y);
Square
Gamma> box = new(Square);
{Square (length) (sides . 4)}
Gamma> class_name(box);
Square
Gamma>

See Also

class_of
**class_of**

*class_of* — gives the class definition of a given instance.

**Synopsis**

```
class_of (instance)
```

**Arguments**

*instance*

An instance of a class.

**Returns**

The class of the *instance*.

**Description**

This function returns the class definition of the *instance*. If the *instance* belongs to a derived class, the most precise class definition is returned (the class which was used to create the instance through a call to *new*).

**Example**

```
  This example is based on the class and method developed in *method*.
```

```
Gamma> class_of (sqB);
( defclass Square RegPolygon [(area . (defun Square.area (self)
                 (sqr (@ self length))))][length (sides . 4)])
Gamma> 
```

**See Also**

*class_name*
defclass

defclass — is the function equivalent of the class statement.

See

class
defmacro, defmacroe

defmacro, defmacroe — are the Lisp equivalents of the macro function.

Synopsis

defmacro (!name, !args, !expression...)
defmacroe (name, args, expression...)

This version of the macro function is only supported by Lisp. See macro.
defun, defune,

defun, defune — are the function equivalents of the function statement.

See

function
defmethod

defmethod — is the function equivalent of the method statement.

See

method
defvar — defines a global variable with an initial value.

Synopsis

defvar (!symbol, value, constant_p?)

Arguments

symbol
A variable name which has not yet been assigned a value.

value
Any s_exp.

constant_p
If non-nil, the symbol will be assigned as a constant.

Returns

The value of the symbol.

Description

This function defines a global variable with an initial value. If the constant_p argument is present and non-nil, then the symbol becomes a constant, and any attempt to set its value in any scope will fail. If the symbol already has a value and constant_p is non-nil or absent, then defvar will return immediately with the current value of the symbol. If constant_p is non-nil and the symbol already has a value, then an error is generated.

The intent of defvar is to provide a value for a symbol only if that symbol has not yet been defined. This allows a Gamma or Lisp file to contain default symbol values which may be overridden before the file is loaded.

Example

Gamma> defvar(a,7,t);
7
Gamma> a;
7
Gamma> a = 5;
Assignment to constant symbol: a
debug l>
Gamma> b = 9;
defvar

Gamma> defvar(b,10);
9
Gamma> b;
9
Gamma>

See Also

set
**destroy**

*destroy* — destroys a class instance.

**Synopsis**

```
destroy (instance)
```

**Arguments**

`instance`

An instance of any class.

**Returns**

`t` when successful, else error.

**Description**

This function destroys instances of classes. When a class instance is destroyed, its data type changes to *destroyed instance* You can test for a destroyed instance by using the predicate `destroyed_p`.

**Example**

```gamma
Gamma> class RegPolygon{sides; length;} (defclass RegPolygon nil [])[length sides] (RegPolygon (length) (sides)) Gamma> destroy (polyA); t Gamma> polyA; #<Destroyed Instance> Gamma> destroyed_p(polyA); t Gamma>
```

**See Also**

`class, new`
eq, equal

eq, equal — compare for identity and equivalence.

Synopsis

```
eq (s_exp, s_exp)
equal (s_exp, s_exp)
```

Arguments

- **s_exp**
  - Any Gamma or Lisp expression.

Returns

- `eq` returns `t` if the two arguments are exactly the same Gamma or Lisp element, otherwise `nil`. `equal` returns `t` if the two arguments "look" the same but are not necessarily pointers to the same memory.

Description

The interpreter's storage mechanism allows a particular element to be referenced from more than one location. Functions like `cons` and `list` do not copy their arguments, but simply construct a higher level entity (in these cases a list) which refers to their arguments. The `copy` function will create a new top-level structure but maintain references to the sub-elements of the original list. The `eq` function tests to see whether two elements are in fact references to the same element. The `equal` function determines whether two elements have identical contents, but are not necessarily references to the same element. All things which are `eq` are also `equal`. Things which are `equal` are not necessarily `eq`.

The `equal` function will travel lists, arrays and instances to compare sub-elements one at a time. The two elements will be `equal` if all of their sub-elements are `equal`. Numbers are compared based on actual value, so that `equal(3, 3.0)` is `t`. Strings are compared using `strcmp`.

Symbols are always unique. A symbol is always `eq` to itself.

Example

```
Gamma> a = #acme;
acme
Gamma> b = #acme;
acme
Gamma> equal(a, b);
```
t
Gamma> eq(a, b);
t
Gamma> a = "acme";
"acme"
Gamma> b = "acme";
"acme"
Gamma> equal(a, b);
t
Gamma> eq(a, b);
nil
Gamma> equal(5, 5);
t
Gamma> eq(5, 5);
nil
Gamma> x = list(#acme, list(1,2,3), "hi");
(acme (1 2 3) "hi")
Gamma> y = copy (x);
(acme (1 2 3) "hi")
Gamma> equal(x, y);
t
Gamma> eq(x, y);
nil
Gamma> equal(cadr(x), cadr(y));
t
Gamma> eq(cadr(x), cadr(y));
t
See Also

Comparison Operators
error

error — redirects the interpreter.

Synopsis

```plaintext
error (string)
```

Arguments

`string`

A string.

Returns

This function does not return.

Description

The `error` function causes the interpreter to immediately stop what it is doing and to jump to the innermost `trap_error`, `unwind_protect`, interactive session or inter-process communication event handler. The value of `_last_error_` is set to the argument string.

Example

This function will return its argument if the argument is a number, or generate an error and never return if the argument is not a number.

```plaintext
function check_number (n)
{
    if (!number_p(n))
    {
        error(string(n," is not a number.");
    }
    n;
}
```

This statement will immediately cause an error if the user presses `Ctrl-C` at the keyboard. This is useful for breaking a running program and going to a debugging prompt.

```plaintext
signal (SIGINT, #(error ("Keyboard Interrupt")))
```

See Also

`trap_error, unwind_protect`
**eval**

`eval` — evaluates an argument.

**Synopsis**

```lisp
eval (s_exp)
```

**Arguments**

- `s_exp`
  - Any Gamma or Lisp expression.

**Returns**

The result of evaluating the argument. Note that the argument is also evaluated as part of the function calling mechanism.

**Description**

The `eval` function forms the basis for running a Gamma program. Every data type has a defined behavior to the `eval` call. These are:

- **symbol** Look up the symbol in the current scope and return the value bound to the symbol. If the symbol is not bound, generate an "Undefined symbol" error.
- **list** Evaluate the first element of the list. If the result is a function, call that function with the rest of the list as arguments. If the first element evaluates to an instance of a class, look up the second element as the method name and resolve that method name in the class or its ancestors. Call the method with the instance bound to self and all other list elements as arguments.
- **all others** All other data types evaluate to themselves.

The `eval` function can be useful when constructing code which must be conditionally executed at a later date, and passed about as data until that time. It may be useful to provide a piece of code as an argument to a generic function so that the function can evaluate it as part of its operation.

**Example**

Note: The `#` operator is used to protect an expression from evaluation. See Quote Operators for more information.

```gamma
Gamma> a = 5;
```

---

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
5
Gamma> b = #a;
a
Gamma> b;
a
Gamma> eval(b);
5
Gamma> 

See Also

eval_list
eval_list

eval_list — evaluates each element of a list.

Synopsis

eval_list (list)

Arguments

list
A list.

Returns

A new list whose elements are the results of evaluating each of the elements of the argument list in turn.

Description

Evaluates each element of the list. Returns the results as a new list whose elements correspond on a one-to-one basis with the elements of the list.

Example

The # operator is used to protect an expression from evaluation. See Quote Operators for more information.

Gamma> a = 5;
5
Gamma> b = 3;
3
Gamma> c = list (#a, #b, "Their sum", #(a+b));
(a b "Their sum" (+ a b))
Gamma> eval_list(c);
(5 3 "Their sum" 8)
Gamma>

See Also

eval
eval_string

eval_string — evaluates a string.

Synopsis

eval_string (string, use_parser?)

Arguments

string
A string.

use_parser

\texttt{t} specifies that the string should be parsed as Gamma code instead of LISP code.

Returns

The result of evaluating the \texttt{string} as if it were a Lisp expression.

Description

This function evaluates a string as if it were a Lisp expression, regardless of whether the file syntax is Gamma or Lisp. This function could be written in Lisp as:

\begin{verbatim}
(defun eval_string (string) (let ((x (parse_string string))) (eval x)))
\end{verbatim}

Example

\begin{verbatim}
Gamma> eval_string("(+ 5 6)");
11
Gamma> testvalue = 75;
75
Gamma> eval_string("testvalue");
75
Gamma>
\end{verbatim}
force, forceq, forceqq

force, forceq, forceqq — assign a value to a symbol, forcing the evaluation of change functions for the symbol.

Synopsis

```
force (symbol, s_exp)
forceq (!symbol, s_exp)
forceqq (!symbol, !s_exp)
```

Arguments

- **symbol**
  - A symbol.
- **s_exp**
  - Any Gamma or Lisp expression.

Returns

The **s_exp** argument.

Description

These functions are identical to the **set**, **setq**, and **setqq**, functions, except in addition to assigning a value to a symbol, and being the functional equivalent of the = (assignment) operator, these functions force Gamma to evaluate the change functions for the symbol even if the value has not changed.

This function is particularly useful when working with DataHub points that contain arrays. Gamma handles arrays from the DataHub by mapping them automatically into Gamma arrays, so you can address individual elements. However, in Gamma, if you have a DataHub array point, represented as $default:myarray, you can modify an element of the array normally, such as $default:myarray[0] = 17; but that does not automatically write back to the DataHub, so nothing gets updated. You have to rewrite the point. Logically you would do this: $default:myarray = $default:myarray; to reassign the point. But this is a null operation since you are just assigning the same value again to the point. Using **force, forceq, forceqq** like this: force($default:myarray, $default:myarray); forces the point change to be sent back to the DataHub.

The **force** function evaluates both of its arguments, **forceq** evaluates only its second argument, and **forceqq** evaluates neither of its arguments. A symbol's value is the value returned as a result of evaluating that symbol. Symbols constitute the Lisp mechanism for representing variables. These functions can only affect the value of a symbol in the current scope.
See Also

Assignment Operators, set,setq, setqq
funcall

funcall — provides compatibility with other Lisp dialects.

Synopsis

\[\text{funcall (function, args)}\]

Arguments

- **function**: A function definition.
- **args**: The arguments to the function.

Returns

The result of the function.

Description

This is provided for compatibility with some other dialects of Lisp. Gamma's version of Lisp, SCADALisp, does not need this function as function definitions can be bound directly to any symbol and called by naming that symbol.

Occasionally this function can be useful in Gamma if a large number of variable arguments are being passed to a function. The called function is named as the first argument and the list of arguments to pass to it are passed as a list in the second arg.

Example

```
Gamma> funcall(atan2, list(5,3));
1.0303768265243125057
Gamma> function plus6 (a,b,c,d,e,f) a+b+c+d+e+f;
(defun plus6 (a b c d e f) (+ (+ (+ (+ a b) c) d) e) f)
Gamma> funcall(plus6, list(1,2,3,4,5,6));
21
```

See Also

defun, function
**function_args**

`function_args` — lists the arguments of a function.

**Synopsis**

```
function_args (function)
```

**Arguments**

`function`

Any function name.

**Returns**

A list of the arguments of `function`.

**Description**

This function lists the arguments of any function. Each argument is returned in the form of an association list, whose first element is the function argument, and whose second element represents the argument modifier(s), if any. The hex numbers that correspond to function modifiers are as follows:

- 0x20000000 Optional (?).
- 0x40000000 Variable length argument (...).
- 0x80000001 Not evaluated (!).

**Example**

```
Gamma> function_args(getprop);
((symbol 0) (property 0))
Gamma> function_args(drain);
((file 0) (t_or_nil 0))
Gamma> function_args(gc);
nil
Gamma> function_args(defvar);
((symbol -2147483648) (value 0) (constant? 536870912))
Gamma> function_args(read);
((file 0))
Gamma> function g (a, b, c) { ((a * b) / c); }
(defun g (a b c) (/ (* a b) c))
Gamma> function_args(g);
((a 0) (b 0) (c 0))
```
See Also

function_body, function_name
**function_body**

function_body — gives the body of a user-defined function.

**Synopsis**

```
function_body (function)
```

**Arguments**

*function*  
Any function.

**Returns**

The function definition in Lisp syntax if the function is user-defined, else `nil`.

**Description**

This function shows the body of a user-defined function in Lisp syntax.

**Example**

```
Gamma> function g(a,b,c) {(a * b)/c};
(defun g (a b c) (/ (* a b) c))
Gamma> function_body(g);
#0=((/ (* a b) c))
Gamma> function h(r,s) {sin(r)/cos(s) * tan(s);}  
(defun h (r s) (* (/ (sin r) (cos s)) (tan s)))
Gamma> function_body(h);
#0=(* (/ (sin r) (cos s)) (tan s))
Gamma> function_body(sort);
nil
Gamma>
```

**See Also**

`function_args, function_name`
function_name

function_name — gives the name of a function.

Synopsis

```
function_name (function)
```

Arguments

```
function
```

Any function.

Returns

The name of the function.

Example

```
Gamma> function grand(a,b,c) {(a * b)/c;}
(defun grand (a b c) ($(a * b) / c))
Gamma> function_name(grand);
grand
Gamma> s = grand;
(defun grand (a b c) ($(a * b) / c))
Gamma> function_name(s);
grand
Gamma> function f () { nil; }
(defun f () nil)
Gamma> function g (x) { princ (function_name(x), "\n"); }
(defun g (x) (princ (function_name x) "\n")
Gamma> g (f);
f
t
Gamma>
```

See Also

function_args, function_body
getprop

getprop — returns a property value for a symbol.

Synopsis

getprop (symbol, property)

Arguments

symbol
A symbol.

property
A symbol naming the property to be fetched.

Returns

The value of the property for the given symbol, or nil if the property is not defined.

Description

Return the value of the property for the given symbol. Once a property has been set for a symbol, it will remain as long as the Gamma program is running.

Example

Gamma> tag001 = 5.5;
5.5
Gamma> setprop(#tag001, #maxlimit,10);
nil
Gamma> getprop(#tag001, #maxlimit);
10
Gamma> getprop(#tag001, #minlimit);
nil
Gamma>

See Also

properties, setprop, setprops
has_cvar

has_cvar — queries for the existence of a class variable.

Synopsis

```lisp
has_cvar (instance|class, variable)
```

Arguments

- `instance|class`
  - An instance of a class; or a class.
- `variable`
  - An class variable name, as a symbol.

Returns

- `t` if any instance, class or any parent (base) class contains the variable, otherwise `nil`.

Description

This function checks for the existence of class variables for a class or instance of a class. It searches all parent (base) classes.

Example

```
This example is based on the class and method developed in method and class_add_cvar. The variable name is preceded by # to prevent evaluation. See Quote Operators for more information.

Gamma> RegPolygon;
#0=(defclass RegPolygon nil [(linethickness . 2)
  (perimeter . (defun RegPolygon.perimeter (self)
      (* (@ self sides) (@ self length))))]
  [length sides])
Gamma> polyD;
(RegPolygon (length) (sides))
Gamma> Square;
(defclass Square RegPolygon [(area . (defun Square.area (self)
      (sqr (@ self length)))]
  [length (sides . 4)])
Gamma> sqB;
(Square (length) (sides . 4))
```
See Also

class_add_cvar
has_ivar

has_ivar — queries for the existence of an instance variable.

Synopsis

```
has_ivar (instance|class, variable)
```

Arguments

- `instance|class`
  
  An instance of a class; or a class.

- `variable`
  
  An instance variable name, as a symbol.

Returns

- `t` if the instance or class contains the named instance variable, or if any parent (base) of the class contains the instance variable, otherwise `nil`.

Description

This function queries an instance or class to determine whether a given instance variable exists for that instance or class. When querying classes, if any parent (base) of that class contains the given instance variable, this function returns `t`. It is possible for a class to contain an instance variable, and an instance of that class not to contain it, but only if `class_add_ivar` was called after the instance was created. See `class_add_ivar` for details.

Example

This example is based on the class and method developed in `method`. The variable name is preceded by `#` to prevent evaluation. See `Quote Operators` for more information.

```lisp
Gamma> Square;
(defclass Square RegPolygon [(area . (defun Square.area (self)
  (sqr (@ self length))))
    [length (sides . 4)]
Gamma> sqB;
{Square (length) (sides . 4)}
Gamma> has_ivar(Square, #sides);
t
Gamma> has_ivar(Square, #perimeter);
```
has_ivar

nil
Gamma> has_ivar(sqB, #area);
nil
Gamma> has_ivar(sqB, #length);
t
Gamma>

See Also

class_add_ivar
**instance_vars**

*instance_vars* — finds all the instance variables of a class or instance.

**Synopsis**

```ruby
instance_vars (instance|class)
```

**Arguments**

*instance|class*

An instance of a class, or a class.

**Returns**

An array of all instance variables defined for the given *instance* or *class*. If an instance is queried, then the values of all instance variables for that instance are also reported.

**Description**

Queries the instance variables of a class or instance.

**Example**

This example is based on the class and method developed in *method*, *class_add_ivar* and *class_add_cvar*.

```ruby
Gamma> polyD;
{RegPolygon (length) (sides)}
Gamma> sqB;
{Square (length) (sides . 4)}
Gamma> instance_vars(RegPolygon);
[length sides]
Gamma> instance_vars(polyD);
[(length) (sides)]
Gamma> instance_vars(Square);
[length (sides . 4)]
Gamma> instance_vars(sqB);
[(length) (sides . 4)]
Gamma>
```

**See Also**

*class, class_add_cvar, class_add_ivar*
is_class_member

is_class_member — checks if an instance or class is a member of a class.

Synopsis

```plaintext
is_class_member (instance|class, class)
```

Arguments

- **instance|class**
  - An instance of a class; or a class.
- **class**
  - A class.

Returns

- *t* if the *instance* or *class* is a member of the *class*, else *nil*.

Description

This function checks if a given instance or class is a member (an instance or derived class) of another class.

Example

This example is based on the classes developed in `class`.

```plaintext
Gamma> sqB = new(Square);
  {Square (length) (sides . 4)}
Gamma> is_class_member(sqB, Square);
  t
Gamma> is_class_member(Square, RegPolygon);
  t
Gamma> is_class_member(sqB, RegPolygon);
  t
Gamma> polyF = new(RegPolygon);
  {RegPolygon (length) (sides)}
Gamma> is_class_member(polyF, Square);
  nil
Gamma>
```
See Also

new
ivar_type

ivar_type — returns the type of a given instance variable.

Synopsis

ivar_type (instance, variable)

Arguments

instance
   A class instance.
variable
   An instance variable name, as a symbol.

Returns

nil if the instance does not contain the variable, or the instance variable type, as assigned by class_add_ivar.

Description

This function returns the instance variable type for a given instance variable. The instance variable type is not used internally by the Gamma or Lisp engine.

Example

Gamma> ivar_type(Osinfo,#cpu);
1

See Also

class_add_ivar
**macro**

**macro** — helps generate custom functions.

**Synopsis**

```
macro name (args) statement
```

**Arguments**

- **name**
  
  The name of the macro.

- **args**
  
  An argument list.

- **statement**
  
  The body of the macro.

**Returns**

A named macro definition in Lisp syntax.

**Description**

This function lets Gamma generate custom functions. The most common type of macro is one that will call different functions for different kinds of arguments. Once the macro has been called on a specific kind of argument, successive calls to the macro for that kind of argument will not be processed by the macro at all, but will be handed straight over to its corresponding function.

One advantage is speed, as the macro code is only executed once. Thereafter only the corresponding function code is executed.

**Example**

1. Define a macro. This macro checks its arguments to see if they are symbols or strings, and performs correspondingly different operations on them.

   ```lisp
   macro symbol_number (!x,!y)
   {
     if (symbol_p(x) && symbol_p(y))
       string(string(x),string(y));
     else if (symbol_p(x) && number_p(y))
       `setq(@x,@y);
     else if (number_p(x) && symbol_p(y))
   }
   ```
2. Calling the macro gives these results:

\[
gamma> \text{symbol\_number}(\text{st,art});
\]
"start"
\[
gamma> \text{symbol\_number}(\text{myvalue,35});
35
Gamma> myvalue;
35
Gamma> symbol\_number(40,yourvalue);
40
Gamma> yourvalue;
40
Gamma> symbol\_number(35,40);
75
Gamma>
\]

3. Define a function that includes the macro, and then call that function.

\[
gamma> \text{function } f(x,y) \{ \text{symbol\_number}(b,3) \};
(defun f (x y) (symbol\_number b 3))
Gamma> f(#r,7);
3
Gamma>
\]

4. Check the function definition. Note that the macro code is now gone. In its place is `setq`, the function it calls for the specified kind of argument.

\[
Gamma> f;
(defun f (x y) (setq b 3))
Gamma>
\]

**See Also**

function
new

new — creates a new instance of a class.

Synopsis

```
new (class)
```

Arguments

class
  The name of an existing class.

Returns

A new instance of the class.

Description

The `new` function creates a new instance of the specified class and initializes any instance variables which have default values associated with them, or assigns them to `nil` if there is no default specified.

An instance is represented by an open brace, followed by the class name, followed by a sequence of dotted pairs (dotted lists of two elements), each containing an instance variable name and a value, followed by a closing brace. Note that `(x . nil)` is the same as `(x)`. For example, an object of the class `PhPoint` would be `{PhPoint (x . 5) (y . 0)}`.

An instance can be destroyed by `destroy`.

Example

```
Gamma> class RegPolygon(sides; length;)
(defclass RegPolygon nil [[][length sides]])
Gamma> class Square RegPolygon {sides = 4;}
(defclass Square RegPolygon [[][length (sides . 4)])
Gamma> polyA = new(RegPolygon);
{RegPolygon (length) (sides)}
Gamma> sqC = new(Square);
{Square (length) (sides . 4)}
Gamma>
```

See Also

class, destroy
**parent_class**

`parent_class` — returns the closest parent (base) of a class or instance.

**Synopsis**

```
parent_class (instance|class)
```

**Arguments**

`instance|class`

An instance of a class; or a class.

**Returns**

The closest parent (base) class of the `instance` or `class`.

**Description**

This function returns the closest (immediate) parent (base) class of the class or instance provided.

**Example**

```lisp
Gamma> class RegPolygon {sides; length;}
(defclass RegPolygon nil [[][length sides]])
Gamma> class Square RegPolygon {sides = 4;}
(defclass Square RegPolygon [[][length (sides . 4)]])
Gamma> class BigSquare Square {length = 30;};
(defclass BigSquare Square [[][(length . 30) (sides . 4)]])
Gamma> polyA = new(RegPolygon);
(RegPolygon (length) (sides))
Gamma> sqC = new(Square);
(Square (length) (sides . 4))
Gamma> bigD = new(BigSquare);
(BigSquare (length . 30) (sides . 4))
Gamma> parent_class(polyA);
nil
Gamma> parent_class(sqC);
(defclass RegPolygon nil [[][length sides]])
Gamma> parent_class(bigD);
(defclass Square RegPolygon [[][length (sides . 4)]])
Gamma> parent_class(Square);
(defclass RegPolygon nil [[][length sides]])
```
parent_class

Gamma> parent_class(BigSquare);
(deffunction Square RegPolygon [][length (sides . 4)])
Gamma>

See Also

class
print_stack

print_stack — prints a Gamma stack.

Synopsis

```
print_stack (file?, stack)
```

Arguments

- `file`  
  The name of a file.
- `stack`  
  The stack you wish to print.

Returns

- `t` if successful, else `nil`.

Description

This function causes Gamma to print a stack, such as `_error_stack_`, `_eval_stack_`, `_jump_stack_`, or `_unwind_stack_`. See Predefined Symbols for more details about these.

Example

```
Gamma> try (2 + nil); catch print_stack(_eval_stack_);
trap_error + print_stack
   t
Gamma>
```

See Also

- Predefined Symbols
properties

properties — should never be used.

Synopsis

```
properties (symbol)
```

Arguments

```
symbol
```

Any symbol.

Returns

The property list for the given symbol.

Description

This function should never be used. Property lists are designed to be handled by the getprop and setprop functions. Property lists may be represented internally by a number of mechanisms, so the type and structure of the return from this function may change at any time.

See Also

```
getprop, setprop, setprops
```
quote, backquote

quote, backquote — correspond to Quote Operators.

Synopsis

```
quote (s_exp)
backquote (s_exp)
```

Arguments

`s_exp`

Any Gamma or Lisp expression.

Returns

The `s_exp`, without evaluation.

Description

These are the functional equivalents of the Quote Operators. `quote` is identical to the `#quote` operator. `backquote` is identical to the `\quote` operator.

See

Quote Operators
require, load

require, load — load files.

Synopsis

```
require (filename)
load (filename)
require_lisp (filename)
load_lisp (filename)
required_file (filename)
_require_path_ (filename)
```

Arguments

`filename`

The name of a file, as a string.

Returns

The name of the file, as a string.

Description

The `require` function loads the named file the first time that it is called. Subsequent calls to `require` with the same filename will simply be ignored. This provides a means for specifying dependencies for applications containing multiple files.

The `load` function loads the named file every time it is called. It attempts to open the named file, read expressions and evaluate them one at a time until the end of file is reached. `load` attempts to find the file by prepending each of the entries in `_require_path_` to the filename. If the file is not found, then `load` appends each of the entries in `_load_extensions_` to the path resulting from concatenating `_require_path_` and filename. If the file is still not found, `nil` is returned. If a different grammar has been defined for the loader, then that grammar will be used to read the file.

`require_lisp` and `load_lisp` operate similarly to `require` and `load`, except they treat any file as a Lisp file. This is helpful when using Lisp libraries with alternate grammars such as Gamma or user-defined grammars.

`required_file` determines which file would be loaded as the result of a call to `require` or `require_lisp`, but does not actually load it. This can be useful in debugging to determine where a particular function or file is coming from.

The pre-defined global variable `_require_path_` contains a list of the paths to be searched to find the specified filename. This variable is initialized to ("" "/usr/cogent/lib"",

```bash
skkynet.com
```
which references the current directory and the standard location for cogent libraries. The list of paths can be augmented with:

```lisp
_require_path_ = cons ("my_directory_name", _require_path_);
```

**Example**

```gamma
Gamma> require("x/myfile.dat");
"x/myfile.dat"
Gamma> require("x/myfile.dat");
t
Gamma> load("x/myfile.dat");
"x/myfile.dat"
Gamma> required_file("x/myfile.dat");
t
Gamma> require_lisp("myfileli.dat");
nil
Gamma>
```

**See Also**

[Loading Files](#)
set, setq, setqq

set, setq, setqq — assign a value to a symbol.

Synopsis

set (symbol, s_exp)
setq (!symbol, s_exp)
setqq (!symbol, !s_exp)

Arguments

symbol
A symbol.
s_exp
Any Gamma or Lisp expression.

Returns

The s_exp argument.

Description

These functions assign a value to a symbol, and are the functional equivalent of the = (assignment) operator. Normally in Gamma the = operator is used for assignment, but these functions give more control over evaluation of symbols and expressions at the point of assignment.

The set function evaluates both of its arguments. setq evaluates only its second argument and setqq evaluates neither of its arguments. The most commonly used of these functions is setq. A symbol’s value is the value returned as a result of evaluating that symbol. Symbols constitute the Lisp mechanism for representing variables. These functions can only affect the value of a symbol in the current scope.

Example

Gamma> setq(y, 6);
6
Gamma> setq(x, #y);
y
Gamma> set(x, 5);
5
Gamma> x;
y
Gamma> y:
5
Gamma>

See Also

Assignment Operators, force
setprop

setprop — sets a property value for a symbol.

Synopsis

```
setprop (symbol, property, value)
```

Arguments

- **symbol**
  The symbol whose property will be set.
- **property**
  A symbol which identifies the property to be set.
- **value**
  The new value of the property.

Returns

The previous value for that property, or `nil` if there was no previous value.

Description

All symbols in Gamma may have properties assigned to them. These properties are not limited by the scope of the symbol, so that a symbol's property list is always global. A property consists of a `(name . value)` pair. Property lists are automatically maintained by `setprop` to ensure that each property name is unique for a symbol. A symbol may have any number of properties. A property for a symbol is queried using `getprop`.

The `symbol` and `property` are normally protected from evaluation when setting properties, using the `#` operator.

Example

```
Gamma> setprop(#weight,#hilimit,1000);
nil
Gamma> setprop(#weight,#hiwarning,950);
nil
Gamma> setprop(#weight,#lowlimit,500);
nil
Gamma> setprop(#weight,#lowwarning,550);
nil
Gamma> getprop(#weight,#hilimit);
1000
```
Gamma> getprop(#weight,#lowwarning);
550
Gamma>

See Also

getprop, properties, setprops
setprops

description — lists the most recent property value settings.

Synopsis

setprops (symbol, properties)

Arguments

symbol
The symbol whose properties will be listed.

properties
Any property.

Returns

A list of properties with their most recent values, as associated pairs.

Description

This function is used to get a list of all the properties and their associated values as
(name, value) pairs. It is called using any of the symbol's properties. The list contains cur-
tent values in order from the most to least recently entered.

The symbol and property are normally protected from evaluation when setting proper-
ties, using the # operator.

Example

Gamma> setprop(#weight,#hilimit,1000);
nil
Gamma> setprop(#weight,#lowlimit,500);
nil
Gamma> setprop(#weight,#warning,950);
nil
Gamma> setprop(#weight,#warning,975);
950
Gamma> setprops(#weight,#hilimit);
((warning . 975) (lowlimit . 500) (hilimit . 1000))
Gamma>

See Also

setprop
**trap_error**

*trap_error* — traps errors in the body code.

**Synopsis**

```
trap_error (!body, !error_body)
```

**Arguments**

- **body**
  - Any Gamma or Lisp expression.
- **error_body**
  - Any Gamma or Lisp expression.

**Returns**

The result of the *body*, unless an error occurs during its evaluation, in which case the result of evaluating the *error_body*.

**Description**

This function traps any errors which occur while evaluating the *body* code. If no error occurs, then *trap_error* will finish without ever evaluating the *error_body*. If an error does occur, *trap_error* will evaluate the *error_body* code immediately and the error condition will be cleared. This is usually used to protect a running program from a piece of unpredictable code, such as an event handler. If the error is not trapped it will be propagated to the top-level error handler where it will cause the interpreter to go into an interactive debugging session.

**Example**

The following piece of code will run an event loop and protect against an unpredictable event.

```plaintext
while(t)
{
    trap_error(next_event(),print_trapped_error());
}

function print_trapped_error ()
{
    princ("Error\n", _error_stack_, "\n occurred...\n");
    princ("Clearing error condition and continuing.\n");
}
```
See Also

`error, unwind_protect, try catch`
unwind_protect

unwind_protect — ensures code will be evaluated, despite errors in the body code.

Synopsis

unwind_protect (!body, !protected_body)

Arguments

body
Any Gamma or Lisp expression.

protected_body
Any Gamma or Lisp expression.

Returns

The result of evaluating the protected_body code. If an error occurs then this function does not return.

Description

This function ensures that a piece of code will be evaluated, even if an error occurs within the body code. This is typically used when an error might occur but cleanup code has to be evaluated even in the event of an error. The error condition will not be cleared by this function. If an error occurs then control will be passed to the innermost trap_error function or to the outer level error handler immediately after the protected_body is evaluated.

Example

The following code will close its file and run a write_all_output function even if an error occurs.

if (fp=open("filename","w"))
{
  unwind_protect(write_all_output(),close(fp));
}

See Also

error, trap_error, protect unwind
whence

whence — gives input information.

Synopsis

whence (s_exp)

Arguments

s_exp

Any Gamma or Lisp expression.

Returns

A list whose car is the input source, and whose cdr is an integer showing the sequential input order of the expression.

Description

This function checks the input source and sequence of any given Gamma expression. It returns this information in the form of a list. The sequence number is assigned the first time the expression is used, and does not change. The whence call itself is a Gamma expression, and thus generates a sequence number each time it is called.

This function requires Gamma to be running in debugging mode. To start debugging mode, you must include the -d option when starting Gamma.

Example

[~/w/devel/lisp]$ gamma -d
Gamma(TM) Advanced Programming Language
Copyright (C) Cogent Real-Time Systems Inc., 1996. All rights reserved.
Version 2.4 Build 147 at Sep 13 1999 17:15:51
Gamma> c = 12;
 12
Gamma> d = 14;
 14
Gamma> whence(c);
("stdin" 1)
Gamma> whence(d);
("stdin" 2)
Gamma> f = 16;
16
Gamma> whence(f);
("stdin" 6)
Gamma>
append

append — concatenates several lists into a single new list.

Synopsis

append (list...)

Arguments

list

One or more lists.

Returns

A new list whose elements are the all of the elements in the given lists, in the order that they appear in the argument lists.

Description

This function concatenates all of the argument lists into a single new list, in the order the arguments are given. Each list is appended to the preceding list by assigning it to the cdr of the last element of that list. The appending is non-destructive; for a destructive version of append use nappend.

Example

Gamma> append (list(1,2,3), list(4,5));
(1 2 3 4 5)
Gamma> append (list(#a,#c,#g), list(#b,#d,#z));
(a c g b d z)
Gamma>

See Also

nappend
aref

aref — returns an array expression at a given index.

Synopsis

```
aref (array, index)
```

Arguments

- `array` - An array.
- `index` - A number giving the index into the array, starting at zero.

Returns

The array element at the given index in the array.

Description

The index starts at zero, and extends to the length of the array minus one. If the index is not valid in the array, `nil` is returned, but no error is generated.

Note: This function is identical to the square bracket syntax for referencing array elements, with the syntax:

```
array[index]
```

Example

```
Gamma> x = array (1, 5, #d, "farm");
[1 5 d "farm"]
Gamma> aref (x, 0);
1
Gamma> aref (x, 8);
nil
Gamma> x[3];
"farm"
Gamma>
```

See Also

`array, aset, delete, insert, sort`
array

array — constructs an array.

Synopsis

array (s_exp?...)

Arguments

s_exp
Any Gamma or Lisp expression.

Returns

An array containing all of the arguments.

Description

An array is represented as a sequence of objects surrounded by square brackets, as in [1 2 a (4 5)]. The objects within the brackets are not evaluated. To refer to or access an array, it must be assigned to a symbol.

This function constructs an array of all of the arguments, in the order given. The arguments are evaluated when the array function is called, but once the array has been constructed the array objects are not evaluated.

It is possible to create an empty array, and fill it later. It will expand as necessary when array objects are added.

Example

Gamma> array(#a, 5, nil, 4 + 3, "goodbye");
[a 5 nil 7 "goodbye"]
Gamma> y = array(5 * 2, #symbol, 432, "string", nil);
[10 symbol 432 "string" nil]
Gamma> z = array(#c, y);
[c [10 symbol 432 "string" nil]]
Gamma> x = array();
[]
Gamma> x[5] = 19;
19
Gamma> x;
[nil nil nil nil nil 19]
See Also

aset, aref, insert, delete, sort
array_to_list

array_to_list — converts an array to a list.

Synopsis

array_to_list (array);

Arguments

array
Any array.

Returns

The array converted to a list.

Description

This convenience function converts the top level of an array to a list. Lower level arrays in the resulting list will remain unchanged unless converted separately.

Example

Gamma> a = array(1,2,3);
[1 2 3]
Gamma> b = array(4,5,6);
[4 5 6]
Gamma> c = array(7,8,9);
[7 8 9]
Gamma> d = array(a,b,c);
[[1 2 3] [4 5 6] [7 8 9]]
Gamma> e = array_to_list(d);
([1 2 3] [4 5 6] [7 8 9])
Gamma> list_p(e);
t
Gamma> list_p(a);
nil
Gamma>

See Also

list_to_array
**aset**

`aset` — sets an array element to a value at a given index.

**Synopsis**

```plaintext
text
```

**Arguments**

- `array`
  - An array.
- `index`
  - A numeric index into the array.
- `value`
  - The new value to be placed in the array.

**Returns**

The `value` argument.

**Description**

Sets an array element to the value at the index. If the index is past the end of the array, then the array will be extended with nils to the index and the value inserted.

This function can also be called using square bracket syntax for referencing array elements, with the syntax:

`array[index]`

**Example**

```plaintext
Gamma> x = array (3, 5, #b, nil);
[3 5 b nil]
Gamma> aset(x, 3, 7);
7
Gamma> x;
[3 5 b 7]
Gamma> x[0] = 9;
9
Gamma> x;
[9 5 b 7]
Gamma>
```
See Also

array, aref, insert, delete, sort
**assoc, assoc_equal**

 assoc, assoc_equal — search an association list for a sublist.

**Synopsis**

```plaintext
assoc (s_exp, list)
assoc_equal (s_exp, list)
```

**Arguments**

- **s_exp**
  
  Any Gamma or Lisp expression.

- **list**
  
  An association list.

**Returns**

A list whose members are the remainder of the association list starting at the element whose car is eq or equal to the *s_exp*.

**Description**

An association list is a list whose elements are also lists, each of which typically contains exactly two elements. *assoc* searches an association list for a sublist whose car is eq to the given *s_exp*. *assoc_equal* uses equal instead of eq for the comparison. If no matching sublist is found, returns *nil*.

A symbol-indexed association list (or sym-alist) is an association list where the car of each element is a symbol. This is a common construct for implementing property lists and lookup tables. Since symbols are always unique, sym-alists can be searched with *assoc* instead of *assoc_equal*.

**Example**

```gamma
Gamma> a = 10;
10
Gamma> b = 20;
20
Gamma> c = 30;
30
Gamma> x = list (list(a,15), list(b,25), list(c, 35));
((10 15) (20 25) (30 35))
Gamma> assoc (b,x);
```
((20 25) (30 35))
Gamma> assoc (20, x);
nil
Gamma> assoc_equal(20, x);
((20 25) (30 35))
Gamma>

See Also

car, cdr, eq, equal, Data Types and Predicates
**bsearch**

*bsearch* — searches an array or list for a element.

**Synopsis**

```
bsearch (list_or_array, key, compare_function)
```

**Arguments**

- `list_or_array`
  A list or array whose elements are sorted.
- `key`
  The list or array element to search for.
- `compare_function`
  A function used to compare the `key` with the array elements.

**Returns**

An association list composed of the `key` and it's position in the array.

**Description**

This function performs a binary search on an array based on a comparison function you provide. The `compare_function` must return a negative number if the value is ordinally less than the list or array element, 0 if the two are equal and a positive number if the value is ordinally greater than the list or array element. The `array` or `list` must be sorted in an order recognizable by the `compare_function` for this function to work.

**Example**

```
Gamma> function comp (x,y) {x - y;}
(defun comp (x y) (- x y))
Gamma> Ax = array(9,2,11,31,13,8,15,95,17,5,19,6,21);
[9 2 11 31 13 8 15 95 17 5 19 6 21]
Gamma> Sx = sort(Ax,comp);
[2 5 6 8 9 11 13 15 17 19 21 31 95]
Gamma> bsearch(Sx,19,comp);
(19 . 9)
Gamma> bsearch(Sx,5,comp);
(5 . 1)
Gamma>
```
See Also

sort
car, cdr, and others

car, cdr, and others — return specific elements of a list.

Synopsis

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>car (list)</td>
<td>Returns the first element of a list.</td>
</tr>
<tr>
<td>cdr (list)</td>
<td>Returns all of the list except for the first element.</td>
</tr>
<tr>
<td>caar (list)</td>
<td>Nesting of car and cdr to return the first two elements.</td>
</tr>
<tr>
<td>cadr (list)</td>
<td>Nesting of car and cdr to return the second element.</td>
</tr>
<tr>
<td>cdar (list)</td>
<td>Nesting of car and cdr to return the second element.</td>
</tr>
<tr>
<td>cddr (list)</td>
<td>Nesting of car and cdr to return the second and third elements.</td>
</tr>
<tr>
<td>caaar (list)</td>
<td>Nested car and cdr to return the first three elements.</td>
</tr>
<tr>
<td>caadr (list)</td>
<td>Nested car and cdr to return the second and third elements.</td>
</tr>
<tr>
<td>cadar (list)</td>
<td>Nested car and cdr to return the second and third elements.</td>
</tr>
<tr>
<td>caddr (list)</td>
<td>Nested car and cdr to return the second, third, and fourth elements.</td>
</tr>
<tr>
<td>cdaar (list)</td>
<td>Nested car and cdr to return the first four elements.</td>
</tr>
<tr>
<td>cdadr (list)</td>
<td>Nested car and cdr to return the second and fourth elements.</td>
</tr>
<tr>
<td>cddar (list)</td>
<td>Nested car and cdr to return the second, third, and fourth elements.</td>
</tr>
<tr>
<td>cdddr (list)</td>
<td>Nested car and cdr to return the first five elements.</td>
</tr>
</tbody>
</table>

Arguments

list

Any list.

Returns

An element of the list or nil.

Description

The car function returns the first element of a list. The cdr function returns all of the list except for the first element. The remaining functions in this group are simply shortcuts for the common combinations of car and cdr. The shortcut functions are read from left to right as nested car and cdr calls. Thus, a call to cadr (mylist) would be equivalent to car (cdr (mylist)). If the argument is not a list, the result is nil. The cdr function will only return a non-list result in the case of a dotted pair.

Example

Gamma> car(list(1,2));
1
Gamma> cdr(list(1,2));
See Also

cons, list, nth_car, nth_cdr
cons

cons — constructs a cons cell.

Synopsis

\[
\text{cons} (\text{car}\_\text{exp}, \text{cdr}\_\text{exp})
\]

Arguments

\text{car}\_\text{exp}
Any Gamma or Lisp expression.

\text{cdr}\_\text{exp}
Any Gamma or Lisp expression.

Returns

A list whose car is \text{car}\_\text{exp} and whose cdr is \text{cdr}\_\text{exp}.

Description

This function constructs a list whose car and cdr are \text{car}\_\text{exp} and \text{cdr}\_\text{exp} respectively. This construction is also known as a cons cell. If the \text{cdr}\_\text{exp} is a list, this has the effect of increasing the list by one member, that is, adding the \text{car}\_\text{exp} to the beginning of the \text{cdr}\_\text{exp}.

Example

\begin{verbatim}
Gamma> a = list(2,3,4);
  (2 3 4)
Gamma> b = 5;
  5
Gamma> cons(b,a);
  (5 2 3 4)
Gamma> cons(a,b);
  ((2 3 4) . 5)
Gamma> cons(5,nil);
  (5)
\end{verbatim}

See Also

Data Types and Predicates
copy

makes a copy of the top list level of a list.

Synopsis

\[
\text{copy (s\_exp)}
\]

Arguments

\(\text{s\_exp}\)

Any Gamma or Lisp expression.

Returns

A copy of the top list level of the argument.

Description

This function makes a copy of the top list level of the argument if the argument is a list, otherwise it simply returns the argument. This produces a new list which is equal to the previous list, and whose elements are eq. That is, the elements are not copied but simply reside in both the original and the copy.

Example

\[
\begin{align*}
\text{Gamma}\>&\ a = \text{list}(1, \text{list}(2,3,\text{list}(4),5)); \\
&\quad (1 \ (2 \ 3 \ (4) \ 5)) \\
\text{Gamma}\>&\ b = \text{copy}(a); \\
&\quad (1 \ (2 \ 3 \ (4) \ 5)) \\
\text{Gamma}\>&\ \text{cadr}(a); \\
&\quad (2 \ 3 \ (4) \ 5) \\
\text{Gamma}\>&\ \text{equal(cadr(a),cadr(b))}; \\
&\quad \text{t} \\
\text{Gamma}\>&\ \text{eq(cadr(a),cadr(b))}; \\
&\quad \text{t} \\
\text{Gamma}\>&\
\end{align*}
\]

See Also

copy_tree, eq, equal
**copy_tree**

*copy_tree* — copies the entire tree structure and elements of a list.

**Synopsis**

```
copy_tree (s_exp)
```

**Arguments**

- **s_exp**
  - Any Gamma or Lisp expression.

**Returns**

A copy of the entire tree structure and elements of the argument, if it is a list. Otherwise, the argument.

**Description**

This function makes a recursive copy of the entire tree structure of the argument if the argument is a list, otherwise it simply returns the argument. This produces a new list which is equal to the previous list, and whose elements are equal, but not eq. That is, the elements are all copied down to the level of the non-list leaves. They are equal to the original elements, but they are different elements. Thus they are equal but not eq.

**Example**

```
Gamma> a = list(1,list(2,3,list(4),5));
(1 (2 3 (4) 5))
Gamma> b = copy_tree(a);
(1 (2 3 (4) 5))
Gamma> cadr(a);
(2 3 (4) 5)
Gamma> equal(cadr(a),cadr(b));
t
Gamma> eq(cadr(a),cadr(b));
nil
Gamma>
```

**See Also**

*copy, eq, equal*
delete

delete — removes an element from an array.

Synopsis

```
delete (array, position)
```

Arguments

- `array`
  - An array.
- `position`
  - An integer giving the zero-indexed position of the array element to delete.

Returns

The deleted array element, or `nil` if none was deleted. The return value will also be `nil` if the deleted element was `nil` itself.

Description

This function removes an element from an `array` and compresses the rest of the array to reduce its overall length by one. If the position is beyond the bounds of the array, nothing happens. This function is destructive.

Example

```
Gamma> a = [1,2,3,4,5];
[1 2 3 4 5]
Gamma> delete(a,3);
4
Gamma> a;
[1 2 3 5]
Gamma>
```

See Also

- `insert`
difference

difference — constructs a list of the differences between two lists.

Synopsis

difference (listA, listB)

Arguments

listA
  A list.
listB
  A list.

Returns

All elements in listA that are not in listB.

Description

Constructs a new list that contains all of the elements in listA not contained in listB as compared by the function eq.

Example

Gamma> a = 1;
1
Gamma> b = 2;
2
Gamma> c = 3;
3
Gamma> d = 4;
4
Gamma> e = 5;
5
Gamma> A = list (a, b, c);
  (1 2 3)
Gamma> B = list (b, d, e, c);
  (2 4 5 3)
Gamma> difference (A, B);
  (1)
Gamma> difference (B, A);
  (4 5)
See Also

eq, equal, intersection, union
find, find_equal

find, find_equal — search a list using the eq and equal functions.

Synopsis

```
fnd (s_exp, list)
find_equal (s_exp, list)
```

Arguments

- `s_exp`
  Any Gamma or Lisp expression.
- `list`
  A list to be searched.

Returns

The tail of the `list` starting at the matching element. If no match is found, `nil`.

Description

The `find` function searches the `list` comparing each element to the `s_exp` with the function `eq`. The `find_equal` function uses `equal` instead of `eq` for the comparison.

Example

```
Gamma> find(#a,#list(d,x,c,a,f,t,l,j));(a f t l j)
Gamma> find("hi", #list("Bob","says", "hi"));
nil
Gamma> find_equal("hi",#list("Bob","says","hi"));
("hi")
```

See Also

`eq`, `equal`
insert

insert — inserts an array value at a given position.

Synopsis

```
insert (array, position|compare_function, value)
```

Arguments

- **array**
  - An array.
- **position**
  - A number giving the zero-based position of the new element within the array, or a function.
- **compare_function**
  - A function on two arguments used to compare elements in the list or array.
- **value**
  - Any Gamma or Lisp expression.

Returns

The value inserted.

Description

The `insert` function widens an array at the given `position` and inserts the `value`. If a `compare_function` is used, it must return a negative number if the value is ordinally less than the array element, 0 if the two are equal and a positive number if the value is ordinally greater than the array element. The `value` will be inserted using a binary insertion sort, based on the return value of the function.

Example

```
Gamma> x = array("a", "b", "c");
["a" "b" "c"]
Gamma> insert(x,3,"d");
"d"
Gamma> x;
["a" "b" "c" "d"]
Gamma> insert(x,strcmp,"acme");
"acme"
Gamma> x;
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
["a", "acme", "b", "c", "d"]

See Also

aref, array, aset
intersection

intersection — constructs a list of all the elements found in both of two lists.

Synopsis

\[
\text{intersection (listA, listB)}
\]

Arguments

- listA
  A list.
- listB
  A list.

Returns

All elements which appear in both listA and listB.

Description

This function generates a new list which contains all of the elements that appear in both listA and listB. The elements are compared using eq. The order of the elements in the resulting list is not defined.

Example

Gamma> A = list(#a,#b,#c);
(a b c)
Gamma> B = list(#b,#c,#d);
(b c d)
Gamma> intersection(A,B);
(b c)
Gamma>

See Also

eq, equal difference union
length

length — counts the number of elements in a list or array.

Synopsis

\[
\text{length (list)}
\]

Arguments

list
A list or array.

Returns

The number of elements in the list or array. If the argument is not a list or array, returns 0.

Example

\[
\begin{align*}
\text{Gamma} > & \quad \text{length(list(#a,#b,#c,#d))} \\
& 4 \\
\text{Gamma} > & \quad \text{length([11,13,15,17,19,21])} \\
& 6 \\
\text{Gamma} > & \quad \text{length(sqr(2 + 3))} \\
& 0 \\
\text{Gamma} > &
\end{align*}
\]
list, listq

list, listq — create lists.

Synopsis

\[
\text{list} (s\_exp\?\ldots) \\
\text{listq} (!s\_exp\?\ldots)
\]

Arguments

\(s\_exp\)

Any Gamma or Lisp expression.

Returns

A list containing all of the arguments.

Description

A list is represented as a sequence of objects surrounded by parentheses, as in (1 2 a [4 5]), possibly with a dot between the second-to-last and last elements in the list. A literal list can be read from a file or from the command line, but must be quoted (using a quote operator) within code to make it literal.

The \texttt{list} function creates a list from its arguments. \texttt{listq} creates a list from its arguments without evaluation.

Example

\begin{verbatim}
Gamma> list(4+5,6,"hi",#xref);
(9 6 "hi" xref)
Gamma> listq(4+5,6,"hi",#xref);
((+ 4 5) 6 "hi" 'xref)
Gamma>
\end{verbatim}

See Also

Data Types and Predicates Lists and Arrays
list_to_array

list_to_array — converts a list to an array.

Synopsis

list_to_array (list)

Arguments

list

A list to convert to an array.

Returns

The list converted to an array.

Description

This convenience function converts the top level of a list to an array. Sub-lists will remain unchanged unless converted separately.

Example

Gamma> a = list(1,2,3);
(1 2 3)
Gamma> b = list(4,5,6);
(4 5 6)
Gamma> c = list(7,8,9);
(7 8 9)
Gamma> d = list(a,b,c);
((1 2 3) (4 5 6) (7 8 9))
Gamma> e = list_to_array(d);
[(1 2 3) (4 5 6) (7 8 9)]
Gamma> array_p(e);
t
Gamma> array_p(a);
nil
Gamma>

See Also

array_to_list
**make_array**

*make_array* — creates an empty array.

**Synopsis**

```
make_array (n_elements)
```

**Arguments**

*n_elements*

A number of elements.

**Returns**

An array with the given number of elements, all `nil`.

**Description**

Creates an empty array for later use. This function has become obsolete as the `array` function can now create empty arrays. See `array`.

**Example**

```
Gamma> make_array(4);
[nil nil nil nil]
Gamma> make_array(7);
[nil nil nil nil nil nil nil]
Gamma>
```

**See Also**

`array`
nappend

nappend — appends one or more lists, destructively modifying them.

Synopsis

nappend (list...)

Arguments

list

One or more lists which will be appended in order.

Returns

The first list, modified in place with the remaining lists appended onto it.

Description

This function appends one or more lists, destructively modifying all but the last argument. It is otherwise identical to append.

Example

Gamma> a = list (1, 2, 3);
(1 2 3)
Gamma> b = list (4, 5, 6);
(4 5 6)
Gamma> c = list (7, 8, 9);
(7 8 9)
Gamma> nappend (a, b, c);
(1 2 3 4 5 6 7 8 9)
Gamma> a;
(1 2 3 4 5 6 7 8 9)
Gamma> b;
(4 5 6 7 8 9)
Gamma> c;
(7 8 9)
Gamma>

See Also

append
**nremove**

*nremove* — removes list items, destructively altering the list.

**Synopsis**

```
nremove (s_exp, list, use_equal?)
```

**Arguments**

- **s_exp**
  
  Any Gamma or Lisp expression.

- **list**
  
  A list.

- **use_equal**
  
  If non-nil, use `equal` instead of `eq` for comparison.

**Returns**

The list with any elements which are `eq` (or `equal` if specified) to `s_exp` destructively removed.

**Description**

This function removes all occurrences of the `s_exp` within the given list and destructively alters the list to reduce its size by one for each occurrence. The default comparison used is `eq`. If the first argument is removed, then the return value will be `(cdr list)` with all other occurrences of `s_exp` destructively removed.

**Example**

```
Gamma> y = list (#a, #b, #c);  
   (a b c)  
Gamma> nremove (#b, y);  
   (a c)  

Gamma> x = list(1,2,3,4,5,6);  
   (1 2 3 4 5 6)  
Gamma> nremove(3, x);  
   (1 2 3 4 5 6)  
Gamma> nremove(3, x, t);  
   (1 2 4 5 6)
```
Gamma> y = list(1,2,3,4,1,2,3,4,1,2);
(1 2 3 4 1 2 3 4 1 2)
Gamma> nremove (1,y,t);
(2 3 4 2 3 4 2)
Gamma>

See Also

nreplace, remove
nreplace, nreplace_equal

nreplace, nreplace_equal — replace elements in a list.

Synopsis

```
nreplace (new_s_exp, old_s_exp, list)
nreplace_equal (new_s_exp, old_s_exp, list)
```

Arguments

- `new_s_exp`  
The new expression to be inserted into the list.
- `old_s_exp`  
The expression in the list to be replaced.
- `list`  
A list.

Returns

The `list`, with all occurrences of `old_s_exp` destructively replaced by `new_s_exp`.

Description

`nreplace` traverses the `list`, replacing any element which is `eq` to `old_s_exp` with `new_s_exp`. `nreplace_equal` uses `equal` as its comparison function.

Example

```
Gamma> R = list (#f, nil, 5, #ftg);
(f nil 5 ftg)
Gamma> nreplace(#h, #ftg, R);
(f nil 5 h)

Gamma> x = list(1,2,3,1,6,7);
(1 2 3 1 6 7)
Gamma> nreplace(4,1,x);
(1 2 3 1 6 7)
Gamma> nreplace_equal(4,1,x);
(4 2 3 4 6 7)
Gamma> x;
(4 2 3 4 6 7)
Gamma>
```
See Also

remove, nremove
nth_car, nth_cdr

nth_car, nth_cdr — iteratively apply the car and cdr functions to a list.

Synopsis

```
nth_car (list, number)
nth_cdr (list, number)
```

Arguments

- **list**
  - Any list.
- **number**
  - The number of cars (or cdrs) to apply to the list argument. Non-integers are rounded down. Non-numbers are treated as zero.

Returns

An element of the list, or nil.

Description

The nth_car and nth_cdr functions iteratively apply the car and cdr functions to a list. If the list argument is not a list, or if the result of any subsequent application of car or cdr is not a list, the result is nil. If the number of applications is less than or equal to 0, the result is the original list.

Example

```
Gamma> c = list (list(list(list(4,5))));
((((4 5))))
Gamma> nth_car(c,2);
((4 5))
Gamma> nth_car(c,4);
4

Gamma> b = list (6,7,8,9,10);
(6 7 8 9 10)
Gamma> nth_cdr (b,2);
(8 9 10)
Gamma> nth_cdr(b,5);
nil
Gamma> nth_cdr(b,4);
```

(10)
Gamma>

See Also

cons, list, car, cdr
**remove**

*remove* — removes list items without altering the list.

**Synopsis**

```
remove (s_exp, list, use_equal?)
```

**Arguments**

- **s_exp**
  - An expression to remove from the list.
- **list**
  - The list from which to remove the s_exp.
- **use_equal**
  - An optional argument. If `t`, *remove* uses the `equal` function for equality, otherwise it uses the more stringent `eq`.

**Returns**

The *list*, with any matching *s_exp* removed.

**Description**

This function non-destructively walks a list and removes elements matching the passed *s_exp* using either `eq` or `equal`.

**Example**

```
Gamma> A = list (#a, #b, #c, #b, #a);
(a b c b a)
Gamma> remove (#a, A);
(b c b)
Gamma> A;
(a b c b a)
Gamma> B = list(1,2,3,2,1);
(1 2 3 2 1)
Gamma> remove(2,B);
(1 2 3 2 1)
Gamma> remove(2,B,t);
(1 3 1)
Gamma>
```
See Also

nremove
reverse

reverse — reverses the order of list elements.

Synopsis

reverse (list)

Arguments

list

A list.

Returns

A new list which whose top-level structure is the reverse of the input list. If the argument is not a list, returns the argument.

Description

None of the elements of the original list is copied. The resulting list contains elements which are eq to the corresponding elements in the original. The original list is not changed.

Example

Gamma> S = list(1,2,3);
(1 2 3)
Gamma> R = reverse (S);
(3 2 1)
Gamma> S;
(1 2 3)
Gamma> car (S);
1
Gamma> caddr(R);
1
Gamma> eq (car (S),caddr(R));
t
Gamma>
rplaca, rplacd

rplaca, rplacd — replace the car and cdr of a list.

Synopsis

rplaca (cons, s_exp)
rplacd (cons, s_exp)

Arguments

list
A list element.
s_exp
Any Gamma or Lisp expression.

Returns

The s_exp, or nil on failure.

Description

These functions destructively alter the form of a list. rplaca modifies the car of a list, effectively replacing the first element. rplacd modifies the cdr of a list, replacing the entire tail of the list. These functions have no meaning for non-lists. To entirely remove the tail of a list, replace the cdr of the list with nil.

Example

Gamma> x = list(1,2,3,4);
(1 2 3 4)
Gamma> rplaca(x,0);
0
Gamma> x;
(0 2 3 4)
Gamma> rplacd(x,list(7,8,9,10,11,12));
(7 8 9 10 11 12)
Gamma> x;
(0 7 8 9 10 11 12)
Gamma>

See Also

car, cdr
**shorten_array**

*shorten_array* — reduces or expands the size of an array.

**Synopsis**

```
shorten_array (array, size)
```

**Arguments**

- **array**
  - The array to shorten or expand.
- **size**
  - The new length for the array.

**Returns**

The resized array.

**Description**

This function reduces or expands the size of an array by cutting off any elements which extend beyond the given size, or by adding nils to the end of the array until the new size is reached. This function is analogous to the C function, `realloc`.

**Example**

```
Gamma> a = array (1,2,3,4,5);
[1 2 3 4 5]
Gamma> shorten_array(a,3);
[1 2 3]
Gamma> shorten_array(a,9);
[1 2 3 nil nil nil nil nil nil]
Gamma>
```

**See Also**

-array, make_array-
sort

sort — sorts a list or array, destructively modifying the order.

Synopsis

sort (list_or_array, compare_function)

Arguments

list_or_array
   A list or an array.

compare_function
   A function on two arguments used to compare elements in the list or array.

Returns

The input list or array, sorted.

Description

This function sorts the list or array in place, destructively modifying the order of the elements. The compare_function must be a function on two arguments which returns: an integer value less than zero if the first argument is ordinally less than the second, zero if the two arguments are ordinally equal, and greater than zero if the first argument is ordinally greater than the second. This function uses the quicksort algorithm.

Example

Gamma> x = list("one","two","three","four","five");
"one" "two" "three" "four" "five"
Gamma> sort(x,strcmp);
"five" "four" "one" "three" "two"
Gamma> x;
"five" "four" "one" "three" "two"
Gamma>
union

union — constructs a list containing all the elements of two lists.

Synopsis

```
union (listA, listB)
```

Arguments

- `listA`: A list.
- `listB`: A list.

Returns

A new list containing all elements in `listA` plus all elements in `listB` which do not appear in `listA`.

Description

The resulting list will not contain duplicate elements from either list. This function uses `eq` for comparisons.

Example

```
Gamma> union (list (#j,#j,#j,#k,#l,#j),list(#k,#k,#l,#m,#n));
(j k l m n)
Gamma> union(list(1,2),list(5,1,2,7));
(1 2 5 1 2 7)
Gamma>
```

See Also

difference, intersection
Strings and Buffers
### bdelete

bdelete — deletes a number of bytes from a buffer.

#### Synopsis

```
bdelete (buffer, position, length?)
```

#### Arguments

- **buffer**
  A buffer.
- **position**
  The position of the first byte to delete. A number between 0 and the length of the buffer minus 1.
- **length**
  An optional number of bytes to delete. The default is 1. A negative number deletes all bytes to the end. A value of 0 does nothing.

#### Returns

The number contained at the specified position in the buffer, or nil if the buffer is undefined at the given position.

#### Description

This function deletes a specified number of bytes from a raw memory buffer. The buffer length does not change as a result of this function. A zero character is placed at the empty position at the end of the buffer, then the buffer is collapsed.

#### Example

```gamma
Gamma> y = buffer (101, 102, 103, 104);
#{efgh}
Gamma> bdelete(y,1,2);
102
Gamma> y;
#{eh\0h}
Gamma>
```

#### See Also

- delete
binser

binser — inserts a value into a buffer.

Synopsis

\[ \text{binser (buffer, position, value)} \]

Arguments

\textit{buffer}

A buffer.

\textit{position}

A number giving the zero-based position of the new element within the buffer, or a function.

\textit{value}

A number which will be cast to an 8-bit signed integer.

Returns

The value inserted.

Description

The binser function inserts the \textit{value} by moving all other values after \textit{position} one space to the right, and removing the last value from the buffer.

If \textit{position} is a function, it is taken to be a comparison function with two arguments. The value will be inserted using a binary insertion sort with the function as the comparison. A comparison function must return a negative number if the value is ordinally less than the buffer element, 0 if the two are equal, and a positive number if the value is ordinally greater than the buffer element.

Example

\begin{verbatim}
Gamma> \text{x = string_to_buffer("Hellothere");}
"Hellothere"
Gamma> \text{binser(x, 5, 32);}
32
Gamma> \text{x;}
"Hello ther"
Gamma>
\end{verbatim}
See Also

buffer
buffer

buffer — constructs a buffer.

Synopsis

buffer (contents?...)

Arguments

contents

Any Gamma or Lisp expression.

Returns

A buffer containing all of the contents.

Description

This function constructs a buffer of all of the arguments, in the order they are given.

A buffer is printed as a sequence of characters (some consoles may not support a character for every entry) surrounded by curly brackets and preceded by a hash sign, such as: #{\n+6ALWbe}. This representation of a buffer cannot be read back in to Gamma, so a symbol must be assigned to a buffer in order to refer to or work with it.

Example

Gamma> bu = buffer (101, 102, 103, 104, 2 * 25, 4 / 82);
#{efgh2\0}
Gamma> shorten_buffer (bu, 2);
#{ef}
Gamma>

See Also

binset, bdelete
**buffer_to_string**

*buffer_to_string* — converts a buffer to a string.

**Synopsis**

```plaintext
buffer_to_string (buffer)
```

**Arguments**

*buffer*

A buffer.

**Returns**

A string representing the contents of the given *buffer* up to the first zero character, or *nil* if the argument is not a buffer.

**Description**

This function converts the *buffer* into a string by treating each element in the buffer as a single character. The first zero character in the buffer terminates the string.

**Example**

```plaintext
Gamma> x = buffer(104,101,108,108,111);
#(hello)
Gamma> buffer_to_string(x);
"hello"
Gamma>
```

**See Also**

*buffer*
format

format — generates a formatted string.

Synopsis

format (format_string, arguments?...)

Arguments

format_string

A string containing format directives and literal text.

arguments

Expressions which will be matched to format directives on a one-to-one basis.

Returns

A string.

Description

Generates a formatted string using directives similar to those used in the "C" printf function. Text in the format_string will be output literally to the formatted string. Format directives consist of a percent sign (%) followed by one or more characters. The following directives are supported:

• a Any Gamma or Lisp expression. The princ_name (the same result as applying the string function on the expression) of a Lisp expression is written to the result string.

• d An integer number. A numeric expression is cast to a long integer and written to the result string. %d is equivalent to %ld.

• f A floating point number. A numeric expression is cast to a long floating point number and written to the result string.

• g A floating point number in "natural" notation. A numeric expression is cast to a long floating point number and written to the result string using the most easily read notation which will fit into the given field size, if any. If no field size is specified, use a notation which minimizes the number of characters in the result.

• s A character string. A string is written to the result string.

The format directive may contain control flags between the % sign and the format type character. These control flags are:

• - Left justify the field within a specified field size.

• + Numbers with a positive value will begin with a + sign. Normally only negative num-
bers are signed.

- " " (A space). Signed positive numbers will always start with a space where the sign would normally be.
- 0 A numeric field will be filled with zeros to make the number consume the entire field width.

Format directives may contain field width specifiers which consist of an optional minimal field width as an integer, optionally followed by a period and a precision specified as an integer. The precision has different meanings depending on the type of the field.

- a The field width option does not apply to this general case. To specify precision on a s_exp, you can convert it to a string and use the $s$ format directives.
- d The precision specifies the minimum number of digits to appear. Leading zeros will be used to make the necessary precision.
- f The precision specifies the number of digits to be presented after the decimal point. If the precision is zero, the decimal point is not shown.
- g The precision specifies the maximum number of significant digits to appear.
- s The precision specifies the maximum number of characters to appear.

Example

```plaintext
Gamma> pi = 3.1415926535;
3.141592653500000541
Gamma> format("PI is %6.3f",pi);
"PI is  3.142"
Gamma> alpha = "abcdefghijklmnopqrstuvwxyz";
"abcdefghijklmnopqrstuvwxyz"
Gamma> format("Alphabet starts: %.10s",alpha);
"Alphabet starts: abcdefghij"
Gamma> x = [1,2,3,4,5,6,7,8,9];
[1 2 3 4 5 6 7 8 9]
Gamma> format("x is: %a",x);
"x is: [1 2 3 4 5 6 7 8 9]"
Gamma> format("x is: %.6s",string(x));
"x is: [1 2 3"
Gamma>
```
**make_buffer**

`make_buffer` — creates a new, empty buffer.

**Synopsis**

```
make_buffer (n_elements)
```

**Arguments**

`n_elements`

The number of elements (bytes) in the buffer.

**Returns**

A new buffer.

**Description**

This function creates a new, empty buffer with `n_elements` number of bytes, all set to zero.

**Example**

```
Gamma> make_buffer(5);
#{\0\0\0\0\0}
Gamma> make_buffer(12);
#{\0\0\0\0\0\0\0\0\0\0\0\0}
Gamma>
```

**See Also**

`buffer`, `buffer_to_string`
open_string

open_string — allows a string to be used as a file.

Synopsis

```plaintext
open_string (string)
```

Arguments

string

A string.

Returns

A pseudo-file that contains the string if successful, otherwise nil.

Description

This function allows a string to be used as a pseudo-file to facilitate reading and writing to a local buffer. All read and write functions which operate on a file can operate on the result of this call. An attempt to write to the string always appends information destructively to the string. Subsequent reads on the string can retrieve this information. A string is always opened for both read and write.

Example

```plaintext
Gamma> s = open_string("Hello there.");
#<File:"String">
Gamma> read_line(s);
"Hello there."
Gamma> s = open_string("Hello there.");
#<File:"String">
Gamma> read(s); Hello
Gamma> read(s); there.
Gamma> read(s); "Unexpected end of file"
Gamma>
```

See Also

close, open, read, read_char, read_double, read_float, read_line, read_long, read_short, read_until, terpri, write, writec
### parse_string

parse_string — parses an input string.

#### Synopsis

```lisp
parse_string (string, use_gamma?=nil, parse_all?=nil)
```

#### Arguments

- **string**: A character string representing either a Lisp expression or a Gamma statement.
- **use_gamma**: An optional argument that defaults to `nil`. If `nil`, the Lisp parser will be used, otherwise the Gamma parser will be used.
- **parse_all**: An optional argument that defaults to `nil`. If `nil`, only the first statement in the string will be parsed, otherwise all statements up to the end of the string will be parsed.

#### Returns

If `parse_all` is `nil`, return the first statement in the string in internal form. If `parse_all` is non-`nil`, return all statements in the string as a list of expressions in internal form. If an error occurs during parsing, this function will throw an error.

#### Description

This function parses the input string using either the Lisp parser or the Gamma parser, and returns either the first complete statement found in the string or all of the statements to the end of the string.

If only the first statement is parsed, the rest of the string is ignored, even if it is invalid. The result is returned in internal form, effectively an executable Lisp representation. Internal form can be passed directly to the `eval` function for evaluation.

If all statements are returned, they are returned in a list, even if there is only one statement in the string. The resulting list can be passed directly to `eval_list`.

#### Example

```lisp
Gamma> a = parse_string("hello");
hello
Gamma> b = parse_string("(cos 5)");
(cos 5)
```
Gamma> \texttt{c = parse_string("( + 5 6 ) (/ 6 3 )");}
(+ 5 6)
Gamma> \texttt{eval(b)};
0.28366218546322624627
Gamma> \texttt{eval(c)};
11
Gamma>

Using optional arguments:

Gamma> \texttt{parse_string("cos(5);", t)};
(cos 5)
Gamma> \texttt{parse_string("cos(5); sin(5);", t)};
(cos 5)
Gamma> \texttt{parse_string("cos(5); sin(5);", t, t)};
((cos 5) (sin 5))
Gamma> \texttt{parse_string("if (x < 1) y = 1; else y = 0;", t)}
(if (< x 1)
 (setq y 1)
 (setq y 0)
)
Gamma>

See Also

eval, eval_string, open_string
raw_memory

raw_memory — tells the amount of memory in use.

Synopsis

raw_memory ()

Arguments

none

Returns

The amount of raw memory in use by the system.

Example

Gamma> raw_memory();
(72462 818)
Gamma> x = 41;
41
Gamma> raw_memory();
(72787 847)
Gamma> x = 55;
55
Gamma> raw_memory();
(73034 871)
Gamma> y = 10;
10
Gamma> raw_memory();
(73359 900)
Gamma>
shell_match

shell_match — compares string text to a pattern.

Synopsis

shell_match (text, pattern)

Arguments

text
  A text string to compare against the given pattern.

pattern
  A shell style pattern.

Returns

  t if the text matches the pattern, otherwise nil.

Description

This function compares the text to the pattern using shell-style wildcard rules. The available patterns are as follows:

- * matches any number of characters, including zero.
- [c] matches a single character which is a member of the set contained within the square brackets.
- [^c] matches any single character which is not a member of the set contained within the square brackets.
- ? matches a single character.
- {xx,yy} matches either of the simple strings contained within the braces.
- \c (a backslash followed by a character) - matches that character.

Example

To get a directory listing of just *.txt files, use:

shell_match(directory("/etc/readme",0,nil),"*.txt");

Gamma> shell_match("hello","?el[a-m]*");
t
Gamma> shell_match("hello","hel{p,m,ga}");
nil
See Also

apropos
shorten_buffer

shorten_buffer — reduces the size of a buffer.

Synopsis

```
shorten_buffer (buffer, n_elements)
```

Arguments

- **buffer**: The buffer to shorten.
- **n_elements**: The number of elements that the buffer is to be reduced to.

Returns

The shortened buffer.

Description

This function reduces the size of a buffer by cutting off any elements which extend beyond the given size. This function is analogous to the C function, `realloc`.

Example

```
Gamma> b = buffer(119,120,121,122);
#{wxyz}
Gamma> shorten_buffer(b,3);
#{wxy}
Gamma>
```

See Also

- `buffer`
- `make_buffer`
strchr, strrchr

strchr, strrchr — search a string for an individual character.

Synopsis

```c
strchr (string, char_as_string)
strrchr (string, char_as_string)
```

Arguments

- `string`  
  Any character string.
- `char_as_string`  
  A string containing the one character to be found.

Returns

The position of the `char_as_string` within the `string`, where the first character is in position zero. If the `char_as_string` is not found in the `string`, returns -1. `strchr` returns the first occurrence of the character in the string. `strrchr` returns the last occurrence of the character in the string.

Description

These functions search a string for an individual character and return the first or last occurrence of that character within the string. The characters within a string are numbered starting at zero.

Example

```c
Gamma> strchr("apple","a");
0
Gamma> strchr("apple","r");
-1
Gamma> strchr("apple","p");
1
Gamma> strrchr("apple pie","p");
6
Gamma>
```

See Also

`strcmp`, `stricmp`, `string_split`, `strlen`, `strrev`, `strstr`, `substr`
**strcmp, stricmp**

**strcmp, stricmp — compare strings.**

**Synopsis**

```c
strcmp (string, string)
stricmp (string, string)
```

**Arguments**

`string`

Any string.

**Returns**

A negative number if the first `string` is ordinally less than the second `string` according to the ASCII character set, a positive number if the first `string` is greater than the second, and zero if the two strings are exactly equal.

**Description**

These functions can be used as comparison functions for `sort` and `insert`. `stricmp` performs the same function as `strcmp`, but alphabetic characters are compared without regard to case. That is, "A" and "a" are considered equal by `stricmp`, but different by `strcmp`.

**Example**

```
Gamma> strcmp("apple", "peach");
-15
Gamma> strcmp("peach", "apple");
15
Gamma> strcmp("Apple","Apple");
0
Gamma> strcmp("Apple","Apple pie");
-32
Gamma> strcmp("Apple","apple");
-32
Gamma> strcmp("Apple","apple");
0
Gamma>
```
See Also

insert, sort, strchr, strrchr
string

string — constructs a string.

Synopsis

string (s_exp...)

Arguments

s_exp

Any Gamma or Lisp expression.

Returns

A string which is the concatenation of all of the arguments.

Description

This function constructs a string by concatenating the \texttt{princ} names of all of the arguments. Any argument that can be evaluated will be. No separation is provided between arguments in the resulting string.

Example

Gamma> string("A list: ",list(#a,#b,#c), " and a sum: ",2 + 3);
"A list: (a b c) and a sum: 5"
Gamma>

See Also

\texttt{format}
**stringc**

_stringc — constructs a string in Lisp-readable form,

**Synopsis**

```lisp
stringc (s_exp...)
```

**Arguments**

- **s_exp**
  
  Any number of expressions.

**Returns**

A string which is the concatenation of all of the arguments.

**Description**

This function is identical to the `string` function, except that the result is produced in a form which is guaranteed to be in Lisp-readable form. This means that special characters within strings and symbols will be escaped appropriately for the reader, and that new-line, form-feed, and tab characters are translated into their \n, \f, and \t equivalents.

**Example**

Gamma> `string(#my, #symbol);`
"mysymbol"
Gamma> `stringc(#my, #symbol);`
"mysymbol"
Gamma> `stringc("A list: ",list(#a,#b,#c), " and a sum: ",2 + 3);
"A list: "(a b c)" and a sum: \"5"
Gamma>

**See Also**

- `string`
**string_file_buffer**

`string_file_buffer` — queries a string file for its internal buffer.

**Synopsis**

```c
string_file_buffer (string_file)
```

**Arguments**

- `string_file`  
  A file which points to an in-memory string, created by a call to `open_string`.

**Returns**

The characters remaining to be read within the string file.

**Description**

This function queries a string file for its internal buffer.

**Example**

```c
Gamma> a = open_string("my false file");
#<File:"String">
Gamma> read_n_chars(a,3);
#{my }
Gamma> string_file_buffer(a);
"false file"
Gamma>
```

**See Also**

`open_string`
string_split

string_split — breaks a string into individual words.

Synopsis

```
string_split (string, delimiters, max_words)
```

Arguments

string
Any character string.

delimiters
A character string containing delimiter characters.

max_words
The maximum number of words to separate.

Returns

A list containing at most \((\text{max\_words} + 1)\) elements, each of which is a string.

Description

This function breaks a string into individual words wherever it finds any one of the characters in the delimiters string. If \(\text{max\_words}\) is zero or less, there is no limit to the number of words which may be generated. If \(\text{max\_words}\) is greater than zero, then at most \(\text{max\_words}\) words will be generated. If there are any characters remaining in the string once \(\text{max\_words}\) words have been generated, then the remaining characters will be returned as the last element in the result list. If delimiters is the empty string, ",", then the input string will be split at any white space.

Example

```
Gamma> string_split("This is a test","",0);
("This" "is" "a" "test")
Gamma> string_split("This is a test"," ",2);
("This" "is" "a test")
Gamma> string_split("This is a test","ie",0);
("Th" "s" "s a t" "st")
Gamma> string_split("12:05:29",:,:,-1);
("12" "05" "29")
Gamma>
```
See Also

`strchr, strrchr, strstr`
string_to_buffer

string_to_buffer — creates a buffer object from a string.

Synopsis

    string_to_buffer (string)

Arguments

    string
        The string to be converted to a buffer.

Returns

    A buffer whose contents are those of the string.

Description

    This function creates a buffer object from a string. The buffer and string are mapped to different memory areas, so that alterations to one do not affect the other.

Example

    Gamma> a = "rhino";
    "rhino"
    Gamma> b = string_to_buffer(a);
    #{rhino}
    Gamma> a = "hippo";
    "hippo"
    Gamma> b;
    #{rhino}
    Gamma>

See Also

    buffer_to_string
strcvt

strcvt — converts the Windows character set of a string.

Synopsis

strcvt (string, from?, to?)

Arguments

string

The string that you need to convert.

from

An optional argument specifying the Windows code page identifier for the local character set. If no value is entered, the default is 0, for your system's code page identifier.

to

An optional argument specifying the Windows code page identifier of the new character set for the string. If no value is entered, the default is 65001, for UTF8.

Returns

The converted string.

Description

This function lets Windows users convert the local character set for a given string into a different character set. In many cases, this function is used to convert the local character set into UTF8, and can thus be run with a single string argument, using the defaults for the from and to arguments. A list of valid Windows code page identifiers for various character sets can be found online in the Microsoft documentation, or by searching on the term "code page identifiers".

In QNX or Linux this function simply returns the string argument.
strlen

strlen — counts the number of characters in a string.

Synopsis

```c
strlen (string)
```

Arguments

`string`

A string.

Returns

The number of characters in the `string`.

Example

```
Gamma> strlen("Hello");
5
Gamma> strlen("How about a cup of coffee?");
26
Gamma>
```

See Also

`length`
strncmp, strnicmp

strncmp, strnicmp — compare two strings and return a numeric result.

Synopsis

```c
strncmp (string1, string2, length)
strnicmp (string1, string2, length)
```

Arguments

- `string1`:
  - The first string.
- `string2`:
  - The second string.
- `length`:
  - The maximum length of the comparison.

Returns

An integer < 0 if `string1` is lexically less than `string2` to the given length; 0 if the two strings are equal up to the given length; and an integer > 0 if `string1` is lexically greater than `string2` up to the given length.

Description

The `strncmp` function compares two strings and returns a numeric result indicating whether the first string is lexically less than, greater than, or equal to the second string. The comparison will carry on for not more than `length` characters of the shorter string. The `strnicmp` function is the case-insensitive version of `strncmp`.

Example

```
Gamma> strncmp("hello","helicopter",4);
3
Gamma> strncmp("hello","help",3);
0
Gamma> strncmp("Hello","help",3);
-32
Gamma> strnicmp("Hello","help",3);
0
Gamma>
```
See Also

strcmp, stricmp
strrev

strrev — reverses the order of characters in a string.

Synopsis

    strrev (string)

Arguments

    string
    A string.

Returns

    A new string which is the reverse of the given string.

Description

    Automatic, full featured, palindrome creator.

Example

    Gamma> strrev("I Palindrone I");
    "I enordnilaP I"
    Gamma> strrev("Madam, I'm adam");
    "mada m'I ,madaM"
    Gamma> strrev("123456789");
    "987654321"
    Gamma> strrev("poor dan is in a droop");
    "poord a ni si nad roop"
    Gamma>

See Also

    strchr, strrchr
**strstr**

`strstr` — finds the location of a given substring.

**Synopsis**

`strstr (stringA, stringB)`

**Arguments**

*stringA*

A string.

*stringB*

A string.

**Returns**

The position of *stringB* within *stringA*, or -1 if *stringA* does not contain *stringB*.

**Description**

This function finds the first complete occurrence of *stringB* within *stringA* and returns the position of the starting character of the match within *stringA*. The first character in *stringA* is numbered zero. If no match is found, -1 is returned.

**Example**

```
Gamma> strstr("Acme widgets","get");
8
Gamma> strstr("Acme widgets","wide");
-1
Gamma>
```

**See Also**

`strchr`, `strrchr`
**substr**

`substr` — returns a substring for a given location.

**Synopsis**

```plaintext
substr (string, start_char, length)
```

**Arguments**

- `string`  
  A string.
- `start_char`  
  The position number of the first character of the substring.
- `length`  
  The length of the substring.

**Returns**

A new string which is a substring of the input `string`.

**Description**

This function returns a substring of the input string starting at the `start_char` position and running for `length` characters. The first character in the string is numbered zero. If `start_char` is greater than the length of the string, the function returns an empty string. If `start_char` is negative, it is indexed from the end of the string. If it is negative and greater than the length of the string, it is treated as zero—the beginning of the string.

If there are fewer characters than `length` in the string, or if `length` is -1, then the substring contains all characters from `start_char` to the end of the string.

**Example**

```plaintext
Gamma> substr("Acme widgets", 7, 3);
"dge"
Gamma> substr("Acme widgets", 9, -1);
"ets"
Gamma> substr("Acme widgets", -7, 4);
"widg"
Gamma> substr("Acme widgets", -30, 4);
"Acme"
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
See Also

strchr, strrchr, string, strstr
tolower

tolower — converts upper case letters to lower case.

Synopsis

tolower (string|number)

Arguments

string
   Any string.

number
   Any number.

Returns

Strings with all letters converted to lower case. Numbers in integer form. Floating point numbers are truncated.

Description

This function converts any upper case letters in a string to lower case. It will also convert numbers to their base 10 integer representation.

Example

Gamma> tolower("Jack works for IBM.");
"jack works for ibm."
Gamma> tolower("UNICEF received $150.25.");
"unicef received $150.25."
Gamma> tolower(5.3);
5
Gamma> tolower(0b0110);
6
Gamma>

See Also

toupper
**toupper**

toupper — converts lower case letters to upper case.

**Synopsis**

toupper (string|number)

**Arguments**

*string*
   Any string.

*number*
   Any number.

**Returns**

Strings with all letters converted to upper case. Numbers in integer form. Floating point numbers are truncated.

**Description**

This function converts any lower case letters in a string to upper case. It will also convert numbers to their base 10 integer representation.

**Example**

```
Gamma> toupper("Jack works for IBM.");
"JACK WORKS FOR IBM."
Gamma> toupper("UNICEF received $150.25.");
"UNICEF RECEIVED $150.25."
Gamma> toupper(5.3);
5
Gamma> toupper(0b0110);
6
Gamma>
```

**See Also**

tolower
Data Type Conversion
bin

bin — converts numbers into binary form.

Synopsis

```
bin (number)
```

Arguments

*number*

Any number.

Returns

An integer number in binary format.

Description

This function casts any number to an integer, and returns it in a binary representation. Floating point numbers are truncated.

Example

```
Gamma> bin(12);
0b1100
Gamma> bin(12.9342);
0b1100
Gamma> bin(0x3b);
0b00111011
Gamma> bin(0o436);
0b000100011110
Gamma>
```

See Also

dec, hex, oct
char

char — generates an ASCII character from a number.

Synopsis

char (number)

Arguments

number

Any number. This is cast to an integer between 0 and 255. Negative numbers are treated as unsigned 2's complement integers.

Returns

A character string with one character which is the character representation of the ASCII value given as the argument.

Description

This function generates the string representation of an ASCII character value.

Example

Gamma> char (65);
"A"
Gamma> char (188);
"¼"
Gamma> char (350.25);
"^"
Gamma> char (-12);
"ô"
Gamma>

See Also

char_val
char_val

char_val — generates a character's numeric value.

Synopsis

char_val (char_as_string)

Arguments

char_as_string
A string.

Returns

The ASCII (numeric) value of the first character in the argument string.

Description

Generates the ASCII (numeric) representation of the first character in a string.

Example

Gamma> char_val ("A");
65
Gamma> char_val ("q");
113
Gamma> char_val ("hope for all");
104
Gamma> char_val ("3");
51
Gamma> char_val ("ô");
-12
Gamma>

See Also

char
dec

dec — converts numbers into base-10 form.

Synopsis

dec (number)

Arguments

number
Any number.

Returns

An integer number in decimal format.

Description

This function casts any number to an integer, and returns it in decimal (base-10) representation.

Example

Gamma> dec(0b1100);
12
Gamma> dec(0x3b);
59
Gamma> dec(45.95);
45
Gamma> dec('A');
65
Gamma>

See Also

bin, hex, oct
hex

hex — converts numbers into hexadecimal form.

Synopsis

hex (number)

Arguments

number

Any number.

Returns

An integer number in hexadecimal format.

Description

This function casts any number to an integer, and returns it in a hexadecimal representation. Floating point numbers are truncated.

Example

Gamma> hex (12);
0xc
Gamma> hex (12.9341);
0xc
Gamma> hex (0b111011);
0x3b
Gamma> hex ('r');
0x72
Gamma>

See Also

bin, dec, oct
**int**

*int* — converts to integer form.

**Synopsis**

```plaintext
int (s_exp)
```

**Arguments**

*s_exp*

Any Gamma or Lisp expression.

**Returns**

An integer representation of the argument.

**Description**

This function converts the argument to an integer. Floating point numbers are truncated. Binaries, hexadecimals and characters convert to decimal integers. In strings, if the first character(s) are numerical, they will be converted to an integer. Otherwise, a string will return zero. All other expression types generate zero.

**Example**

```
Gamma> int(5.5);
5
Gamma> int(0xc);
12
Gamma> int(0b111011);
59
Gamma> int('h');
104
Gamma> int("63 hello");
63
Gamma> int("hello 63");
0
Gamma> int(random());
0
```

**See Also**

* Literals
number

text — attempts to convert an expression to a number.

Synopsis

number (s_exp)

Arguments

s_exp
Any Gamma or Lisp expression.

Returns

A numeric representation of the s_exp if possible, otherwise zero.

Description

The function attempts to convert its argument to a number. Integer and floating point
values remain untouched. String arguments are converted to numbers by attempting to
read a number from the string starting at the first character in the string. The longest legal
number at the beginning of the string is used. All other data types return zero. If possible,
the result will be an integer. If the result cannot be represented as an integer, a real (float-
ing point) number is returned.

Example

Gamma> number(5);
5
Gamma> number("5.4m");
5.4000000000000003553
Gamma> number("m5.4");
0
Gamma> number(#a);
0
Gamma>
oct

oct — converts numbers into octal form.

Synopsis

oct (number)

Arguments

number

Any number.

Returns

An integer number in octal format.

Description

This function casts any number to an integer, and returns it in an octal representation. Floating point numbers are truncated.

Example

Gamma> oct (12);
0o14
Gamma> oct (12.86223);
0o14
Gamma> oct (0x3b);
0o73
Gamma> oct (0b0101101);
0o55
Gamma>

See Also

bin, dec, hex
symbol
symbol — constructs a symbol from a string.

Synopsis

symbol (string)

Arguments

string
A string.

Returns

A symbol.

Description

This function constructs a symbol whose name is the same as the string, and places that symbol into the symbol table. Subsequent calls to this function with the same string will result in the same symbol, preserving the uniqueness of the symbol. Special characters may be included in the symbol name.

Example

Gamma> symbol("Strange symbol");
Strange\ symbol
Gamma> Strange\ symbol;
5
Gamma> symbol(string("item",2+3));
item5
Gamma>
Math
acos, asin, atan, atan2

acos, asin, atan, atan2 — perform trigonometric arc functions.

Synopsis

acos (number)
asin (number)
atan (number)
atan2 (number, number)

Arguments

number

Any integer or real number. Non-numbers are treated as zero.

Returns

The result of the arc trigonometric function in radians.

Description

These functions perform the arc trigonometric functions arc cosine, arc sine, arc tangent, and arc tangent with 2 arguments. The atan2 function is equivalent to:

atan( y / x );

except that atan2 is able to correctly handle x and y values of zero.

Example

Gamma> acos (0.5);
1.0471975511965978534
Gamma> asin (0.5);
0.52359877559829892668
Gamma> atan (2);
1.107148717794090409
Gamma> atan2 (1, 2);
0.46364760900080609352
Gamma> atan2 (1, 0);
1.570796326794896558
Gamma> atan2 (0, 2);
0
Gamma>
acos, asin, atan, atan2

See Also

sin, cos, tan
and, not, or

and, not, or — are the same as the corresponding Logical Operators.

Synopsis

- `and (!condition[,!condition]...)`
- `not (condition)`
- `or (!condition[,!condition]...)`

Arguments

- `condition`
  Any Gamma or Lisp expression.

Returns

- Non-`nil` or `nil`.

Examples

- `Gamma> not(6);`
  `nil`
- `Gamma> not(nil);`
  `t`
- `Gamma> and(5<6,string("hi ","there"));`
  "hi there"
- `Gamma> and(5>6,string("hi ","there"));`
  `nil`
- `Gamma> x = 5;`
  `5`
- `Gamma> y = 6;`
  `6`
- `Gamma> or(x == 3, y == 0);`
  `nil`
- `Gamma> or(x == 3, y == 6);`
  `t`
See Also

Logical Operators
**band, bnot, bor, bxor**

- `band` — bitwise AND
- `bnot` — bitwise NOT (inversion of all bits in a 32-bit word)
- `bor` — bitwise OR
- `bxor` — bitwise exclusive OR (XOR)

**Synopsis**

```
band (number, number)
bnot (number)
bor (number, number)
bxor (number, number)
```

**Arguments**

- `number` — Any number. Non-numbers are treated as zero.

**Returns**

An integer which is the result of the particular operation.

**Description**

The binary operations cast their arguments to integers, and then perform bitwise operations to produce an integer result.

- `band` bitwise AND
- `bnot` bitwise NOT (inversion of all bits in a 32-bit word)
- `bor` bitwise OR
- `bxor` bitwise exclusive OR (XOR)

**Example**

```
Gamma> band (7,5);
5
Gamma> bnot (7);
-8
Gamma> bor (7,5);
7
Gamma> bxor (7,5);
2
Gamma>
```
See Also

Bitwise Operators
ceil

ceil — rounds a real number up to the next integer.

Synopsis

```plaintext
ceil (number)
```

Arguments

`number`

Any number. Non-numbers are treated as zero.

Returns

The smallest integer that is greater than or equal to the `number`.

Description

This function has the effect of rounding real numbers up to the next integer. Integers are unaffected.

Example

```plaintext
Gamma> ceil(1.1);
2
Gamma> ceil(-1.1);
-1
Gamma> ceil(4);
4
```

See Also

`floor`, `round`
cfand, cfor

cfand, cfor — perform and and or functions with confidence factors.

Synopsis

```
cfand (...)
cfor (...)
```

Arguments

condition

Any Gamma or Lisp expression.

Returns

A confidence factor, an integer between 0 and 100.

Description

These functions determine the confidence factor of one or more expressions. `cfand` returns the lowest confidence factor among all of the passed `condition`s, while `cfor` returns the highest confidence factor among the `condition`s.

Example

```
Gamma> a = 3;
3
Gamma> b = 4;
4
Gamma> set_conf(a,50);
50
Gamma> set_conf(b,10);
10
Gamma> cfand(a,b);
10
Gamma> cfor(a,b);
50
Gamma>
```

See Also

`conf`
conf, set_conf

conf, set_conf — query and set confidence factors.

Synopsis

conf (s_exp)
set_conf (s_exp, number|s_exp)

Arguments

s_exp
   Any Gamma or Lisp expression.

number|s_exp
   Any number, or any expression that evaluates to a number. Non-numbers are treated as zero.

Returns

The confidence factor of the number or s_exp.

Description

All Gamma and Lisp expressions in Gamma have an associated confidence factor between 0 and 100 which may be queried using the conf function. This is typically 100, or fully confident. Exceptions arise only when the user explicitly sets the confidence to another value, or when the DataHub provides a confidence value to the interpreter. The set_conf function will set the confidence of an expression to any numerical value, though legal values are between 0 and 100. Numbers less than 0 indicate indeterminate confidence. Numbers greater than 100 will produce strange results.

Example

Gamma> x = 3;
   3
Gamma> set_conf(x, 40);
   40
Gamma> conf(x);
   40
Gamma>
cos, sin, tan

cos, sin, tan — perform trigonometric functions.

Synopsis

\[
\begin{align*}
\text{cos} & \ (\text{number}) \\
\text{sin} & \ (\text{number}) \\
\text{tan} & \ (\text{number})
\end{align*}
\]

Arguments

\textit{number}

Any number in radians. Non-numbers are treated as zero.

Returns

The result of the trigonometric functions cosine, sine and tangent.

Example

Gamma> \text{cos}(8); \\
-0.14550003380861353808 \\
Gamma> \text{sin}(0.8); \\
0.71735609089952279138 \\
Gamma> \text{tan}(0.5); \\
0.54630248984379048416

See Also

asin, acos, atan, atan2
div

div — divides, giving an integer result.

Synopsis

\[ \text{div (number, number)} \]

Arguments

\[ \text{number} \]

Any number.

Returns

The integer result of the division of the first argument by the second.

Description

This function is equivalent to \text{floor (number/number)}

Example

\[ \text{Gamma}> \text{div(12,5)}; \]
\[ 2 \]
\[ \text{Gamma}> \text{div(23423,899)}; \]
\[ 26 \]
\[ \text{Gamma}> \]

See Also

\text{Arithmetic Operators}
exp

exp — calculates an exponent of the logarithmic base (e).

Synopsis

exp (number)

Arguments

number
Any number.

Returns

The natural logarithmic base, e, raised to the power of the number.

Example

Gamma> exp(0);
1
 Gamma> exp(3);
20.085536923187667924
 Gamma>
floor

floor — rounds a real number down to its integer value.

Synopsis

```plaintext
floor (number)
```

Arguments

`number`

Any number. Non-numbers are treated as zero.

Returns

The largest integer which is less than or equal to the `number`.

Example

```plaintext
Gamma> floor(1.2);
1
Gamma> floor(1.9);
1
Gamma> floor(-1.2);
-2
Gamma> floor(-1.9);
-2
Gamma>
```

See Also

`ceil`, `round`
log, log10, logn

log, log10, logn — calculate logarithms.

Synopsis

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log (number)</td>
<td>Calculate the natural logarithm of the number.</td>
</tr>
<tr>
<td>log10 (number)</td>
<td>Calculate the base 10 logarithm of the number.</td>
</tr>
<tr>
<td>logn (base, number)</td>
<td>Calculate the logarithm of the number in the given base.</td>
</tr>
</tbody>
</table>

Arguments

- **number**: Any numeric value.
- **base**: The logarithmic base.

Returns

- For `log`, the natural logarithm of the argument.
- For `log10`, the base 10 logarithm of the argument.
- For `logn`, the logarithm of the number in the given base.

Description

Non-numeric arguments are treated as zero. Illegal values for the arguments will cause an error.

Example

```plaintext
Gamma> log(2);
0.69314718055994528623
Gamma> log10(2);
0.30102999566398119802
Gamma> logn(8,2);
2.9999999999999995559
Gamma>
```
isnan

isnan — identifies invalid or illegal numbers.

Synopsis

```latex
isnan (number)
```

Arguments

- `number` A floating point number.

Returns

- `t` if the argument is not a number (nan), otherwise `nil`.

Description

This function tests to see whether the argument is a special floating point number that indicates the specific case of an illegal number or an infinite number. `NAN` is a floating point number that represents an invalid state, and is represented in Gamma as a constant. Please refer to Literals for more information on `NAN`.

Example

```plaintext
Gamma> isnan(sqrt(-1));
t
Gamma> isnan(sqrt(2));
nil
```
isinf

isinf — determines if a number is infinite.

Synopsis

isinf (number)

Arguments

number

A floating point number.

Returns

1 if the number is positive and infinite, −1 if it is negative and infinite, otherwise nil.

Example

These examples use the constant INF, which represents an infinite number in Gamma.

Gamma> isinf(95);
nil
Gamma> isinf(INF);
1
Gamma> isinf(-INF);
-1
neg

neg — negates.

Synopsis

neg(number)

Arguments

number

Any number.

Returns

The negative of the number.

Example

Gamma> neg(5);
-5
Gamma> neg(-5);
5
Gamma>
**pow**

pow — raises a base to the power of an exponent.

**Synopsis**

```plaintext
pow (base, exponent)
```

**Arguments**

- **base**
  - Any number.
- **exponent**
  - Any number.

**Returns**

The result of raising the `base` to the given `exponent`.

**Description**

Calculates a base to the power of an exponent. Non-numbers are treated as zero.

**Example**

```
Gamma> pow(2, 3);
  8
Gamma> pow(12, 2);
  144
Gamma> pow(5.2, 4.75);
  2517.7690015606849556
Gamma>
```
random

random — generates random numbers from 0 to 1.

Synopsis

random ()

Arguments

none

Returns

A floating point random number which is greater than or equal to 0 and is less than 1.

Description

This function uses a pseudo-random number generator to generate a non-repeating sequence of numbers randomly distributed across the range of 0 <= x < 1.

The random number generator should be seeded prior to being called by using the set_random function. If the same seed is given to set_random, the same random sequence will result every time.

Example

#!/usr/local/bin/gamma -d

//Seed random number generator to clock setting:
set_random(clock());

//Randomly generate an integer from one to six:
function one_to_six ()
{
   floor(6 * random()) + 1;
}

//Print the results:
x = one_to_six();
princ(x, ";n");

See Also

set_random
round

round — rounds a real number up or down to the nearest integer.

Synopsis

    round (number)

Arguments

    number
    A number.

Returns

    The nearest integer to the number.

Description

    This function rounds its argument to the nearest integer. Values of .5 are rounded up to the next highest integer.

Example

    Gamma> round(8.73);
    9
    Gamma> round(2.21);
    2
    Gamma> round(5.5);
    6
    Gamma> round(5.49);
    5
    Gamma>

See Also

    ceil, floor
set_random

set_random — starts random at a different initial number.

Synopsis

```
set_random (integer_seed)
```

Arguments

`integer_seed`

Any integer number.

Returns

t

Description

This function seeds the random number generator to start the pseudo-random sequence at a different number. The same `integer_seed` will always produce the same pseudo-random sequence. `set_random` is commonly called with an unpredictable `integer_seed`, such as the result of `clock`.

Example

```
Gamma> set_random(95);
t
Gamma> random();
0.26711518364027142525
Gamma> random();
0.8748339582234621048
Gamma> random();
0.30958001874387264252
Gamma> set_random(clock());
t
Gamma> random();
0.41952831624075770378
Gamma> random();
0.99278739839792251587
Gamma> random();
0.42997436970472335815
```
See Also

random
sqr

sqr — finds the square of a number.

Synopsis

sqr (number)

Arguments

number
Any number.

Returns

The square of the number.

Example

Gamma> sqr(11);
121
Gamma> sqr(32.73);
1071.2528999999997268
Gamma>

See Also

sqr

cqrt
sqrt

sqrt — finds the square root of a number.

Synopsis

sqrt (number)

Arguments

number

Any number.

Returns

The square root of the number.

Example

Gamma> sqrt(9);
3
Gamma> sqrt(144);
12
Gamma> sqrt(95);
9.7467943448089631175
Gamma>

See Also

sqr
close

close — closes an open file.

Synopsis

close (file)

Arguments

file
A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

t if the file had been open and was closed successfully, else nil.

Description

This function closes a previously opened file. It is not strictly necessary, as the file will be closed when the garbage collector recognizes that there are no references to the file, but it is extremely good policy. This function will close a string opened for reading and writing as well.

Example

Gamma> fp = open("myfile.dat","r");
#<File:"myfile.dat">
Gamma> close(fp);
t
Gamma> fp;
#<Destroyed Instance>
Gamma> 

See Also

fd_close, open, open_string
fd_close

fd_close — closes a file identified by a file descriptor.

Synopsis

\[ \text{fd_close}\ (fd) \]

Arguments

\( fd \)
A file descriptor as returned from \text{fd_open}.

Returns

\( t \), if successful, otherwise \( \text{nil} \).

Description

This function closes a file identified by a file descriptor, i.e. that was opened by \text{fd_open}.

Example

```lisp
Gamma> \text{require_lisp("const/filesys")};
"const/filesys"
Gamma> \text{fp} = \text{fd_open}("/fd/tty8",O_WRONLY);
4
Gamma> \text{fd_write(fp,"\nHello\n");}
8
Gamma> \text{fd_close(fp)};
t
Gamma> \text{fd_close(fp)};
\text{nil}
```

See Also

close, fd_open, Referencing Files
**fd_data_function**

*fd_data_function* — attaches a write-activated callback to a file.

**Synopsis**

```c
fd_data_function (fd|file, code)
```

**Arguments**

- **fd|file**
  A file descriptor as returned from *fd_open*, or the name of a file pointer to a file that was opened by a call to *open* or *open_string*.

- **code**
  Any valid Gamma program, executable code block, or statement.

**Returns**

The return value of the executed *code*.

**Description**

This function acts as a callback, causing the *code* to execute whenever data is written to the file associated with the *fd* or *file* pointer.

**See Also**

*fd_open, fd_eof_function, fd_write, open, open_string, write*
fd_eof_function

fd_eof_function — attaches an eof-activated callback to a file.

Synopsis

```
fd_eof_function (fd|file, code)
```

Arguments

- **fd|file**
  A file descriptor as returned from `fd_open`, or the name of a file pointer to a file that was opened by a call to `open` or `open_string`.

- **code**
  Any valid Gamma program, executable code block, or statement.

Returns

The return value of the executed `code`.

Description

This function acts as a callback, causing the `code` to execute whenever the end of the file (_eof_) is reached during a call to `fd_read` or one of the other `read` functions. The `fd|file` argument identifies the file.

See Also

`fd_open, fd_data_function, fd_write, open, open_string, write`
fd_open

fd_open — opens a file or device and assigns it a file descriptor.

Synopsis

fd_open (name, mode)

Arguments

name
The name of a file, as a string.

mode
The mode for opening the file.

Returns

A non-negative integer representing the lowest numbered unused file descriptor if success-ful. If an error occurs, the function returns -1 and sets the errno.

Description

This function opens a file for reading and/or writing, and assigns it a file descriptor which is used as an argument by other functions such as fd_read and fd_write. The file that is opened could be a regular file, a directory, or a block or character device. Legal mode values are:

- O_RDONLY  Read-only mode
- O_WRONLY  Write-only mode
- O_RDWR    Read-Write mode

Any combination of the following flags may be bitwise OR-ed with the open mode to modify how the file is accessed:

- O_APPEND  Append (writes guaranteed at the end)
- O_CREAT   Opens with file create
- O_EXCL    Exclusive open
- O_NOCTTY  Don't assign a controlling terminal
- O_NONBLOCK Non-blocking I/O
- O_TRUNC   Open with truncation
- O_DSYNC   Data integrity synch
• **O_SYNC** File integrity synch
• **O_TEMP** Temporary file, don't put to disk
• **O_CACHE** Cache sequential files too

If an error occurs -1 is returned and errno is set to one of the following:

• **EACCESS** Search permission denied on a portion of the path prefix, or the file exists and the permissions required to open the file in the given mode so not exist.
• **EBADFSYS** The file or the path prefix to the file was found to be corrupted
• **EBUSY** The file is already open for writing.
• **EEXIST** O_CREAT and O_EXCL are set and the named file exists
• **EINTR** The function was interrupted by a signal
• **EISDIR** The named file is a directory
• **EMFILE** Too many file descriptors are currently in use by this process
• **ENAMETOOLONG** The length of the path to the file is too long.
• **ENFILE** Too many files are currently open on the system
• **ENOENT** O_CREAT is not set and the file does not exist
• **ENOSPC** The directory or file system which would create the new file cannot be extended
• **ENOTDIR** A component of the path to the file is not a directory
• **ENXIO** O_NONBLOCK is set, the file is a FIFO, O_WRONLY is set, and no process has the file open for reading
• **EROFS** The named file resides on a read-only file system.

**Example**

```
Gamma> require_lisp("const/filesys");
"/usr/cogent/lib/const/filesys.lsp"
Gamma> ptr = fd_open("/fd/ttyp8",O_WRONLY);
4
Gamma> fd_write(ptr,"\nhello\n");
7
```

**See Also**

*fd_close, fd_data_function, fd_eof_function, fd_read, fd_write, ser_setup,*

*Referencing Files*
fd_read

    fd_read — reads a buffer or string from a file identified by a file descriptor.

Synopsis

    fd_read (fd, buffer|string, length?, offset?)

Arguments

    fd
      A file descriptor as returned from fd_open.
    buffer|string
      A buffer or string to be read from the file.
    length
      An integer specifying the length of the buffer or string.
    offset
      An integer specifying the position in the file to begin reading the buffer or string.

Returns

    The number of bytes actually read from the file, or -1 on failure and the errno is set.

Description

    This function reads a buffer or string from the specified file.
    
    When an error occurs, the following errnos are possible:
    
    • EAGAIN The O_NONBLOCK flag is set for the fd and the process would be delayed in
      the read operation.
    • EBADF The passed fd is invalid or not open for writing.
    • EFBIG File is too big.
    • EINTR Read was interrupted by a signal.
    • EINVAL iovcnt was less than or equal to 0, or greater than UIO_MAXIOV.
    • EIO Physical I/O error.

Example

    Gamma> x = fd_open("/fd/ser1",O_RDWR);  
    4
See Also

fd_close, fd_open, fd_read, ser_setup, Referencing Files
fd_to_file

fd_to_file — creates a file pointer from a descriptor.

Synopsis

fd_to_file (fd, mode)

Arguments

fd

A file descriptor as returned from fd_open.

mode

A string indicating the mode for the file: "r" for read-only, "w" for writable, "a" for append.

Returns

t, if successful, otherwise nil.

Description

This function creates a file pointer from a file descriptor.

See Also

fileno, Referencing Files
**fd_write**

*fd_write* — writes a buffer or string to a file identified by a file descriptor.

**Synopsis**

```c
fd_write (fd, buffer|string, length?, offset?)
```

**Arguments**

- **fd**
  A file descriptor as returned from *fd_open*.
- **buffer|string**
  A buffer or string to write to the file.
- **length**
  An integer specifying the length of the buffer or string.
- **offset**
  An integer specifying the position in the file to begin writing the buffer or string.

**Returns**

The number of bytes actually written to the file, or -1 on failure and the errno is set.

**Description**

This function writes a buffer or string to the specified file.

When an error occurs, the following errnos are possible:

- **EAGAIN** The O_NONBLOCK flag is set for the *fd* and the process would be delayed in the write operation.
- **EBADF** The passed *fd* is invalid or not open for writing.
- **EFBIG** File is too big.
- **EINTR** Write was interrupted by a signal.
- **EINVAL** `iovcnt` was less than or equal to 0, or greater than `UIO_MAXIOV`.
- **EIO** Physical I/O error.
- **ENOSPC** No free space remaining on drive.
- **EPIPE** Attempt to write to a pipe (or FIFO) that is not open for write. SIGPIPE is also sent to process.
Example

Gamma> x = fd_open("/fd/ser1",O_RDWR);
4
Gamma> fd_write(x,"hello\n");
6
Gamma> fd_close(x);

See Also

fd_close, fd_open, fd_read, ser_setup, Referencing Files
fileno

fileno — creates a file descriptor from a pointer.

Synopsis

```
fileno (file)
```

Arguments

-file
   A file pointer as returned from open.

Returns

- t, if successful, otherwise nil.

Description

This function creates a file descriptor from a file pointer.

See Also

- fd_to_file, Referencing Files
ioctl

ioctl — performs control functions on a file descriptor.

Synopsis

```c
ioctl (fd, request, value)
```

Arguments

- **fd**
  - A file descriptor as returned from `fd_open`.
- **request**
  - One of the functions listed below in Description.
- **value**
  - A number that supplies additional information needed by the `request` function.

Returns

The return value of the `request` function.

Description

This function performs an ioctl call (C library ioctl subroutine) for the given `fd` file descriptor and `request`. The Gamma ioctl function currently only supports `requests` that take numeric arguments, ie. `value` must be a number. You may make operating-system specific ioctl calls by giving a numeric value for the `request` argument.

The currently supported `requests` are:

- TCSBRK
- TCXONC
- TCFLSH
- TIOCHPCL
- TIOCSEXCL
- TIOXXNCL
- TIOCFLUSH
- TIOCDDRAIN
- TIOCSCTTY
- TIOCMMGET
- TIOCMIBIC
- TIOCMIBIS
- TIOCSET
- TIOCSTART
- TIOCSTOP
- TIOCNOTTY
- TIOCOUTQ
- TIOCSPGRP
- TIOCGPGRP
- TIOCCDIR
- TIOCSDIR
- TIOCCBRK
- TIOCSBRK
- TIOCLGET
- TIOCSETPGRP
- TIOCGETPGRP
- FIONCLEX
- FIONCLEX
- FIONSETOWN
- FIOASYNC
- FIONBIO
- FIONREAD
- SIOCSHIWAT
- SIOCWHIWAT
- SIOCSPGRP
- SIOCSPGPRP
open

open — attempts to open a file.

Synopsis

open (filename, mode, use_parser?)

Arguments

filename
A filename (possibly including the path), as a string.

mode
A string indicating the mode for the file: "r" for read-only, "w" for writable, "a" for append. Other characters might be available, depending on the operating system. Please refer to the documentation for the operating system's `fopen` function.

use_parser
Assume Lisp grammar regardless of the default grammar.

Returns

A file pointer, or nil if the request failed.

Description

This function attempts to open a file. If the file is opened for write ("w"), any previously existing file of the same name will be destroyed. If the file is opened for append ("a") then a previously existing file will be lengthened with subsequent writes, but the data in that file will not be damaged. A file can only be opened read-only ("r") if it already exists. The result of this function may be used as an argument to a variety of read and write operations.

When Gamma opens or creates a file, it creates an abstract file pointer. A printed representation of the file pointer looks like this: #<File:filename>. This representation cannot be read back in to Gamma, and so a symbol must be assigned to the file pointer in order to refer to or work with a file. In common language, we refer to this symbol as the file pointer. For instance, in the examples below, we would say the symbol fp is the file pointer. (See also Referencing Files.)

If use_parser is non-nil, then a call to read will parse the file according to its default grammar. If use_parser is nil, then a call to read will parse the file as if it were a Lisp expression. A file must be opened in Lisp format in order to use calls to read_char, read_double, read_float, read_line, read_long, read_short and read_until.
Examples

An input file contains the following:

```
(setq y 5)
```

Calling `open` will produce:

```
Gamma> fp = open ("myopenfile.dat", "r", nil);
#<File:"myopenfile.dat">  
Gamma> princ(read_line(fp), "\n");
(setq y 5)
  t  
Gamma> fp = open ("myopenfile.dat", "r", nil);
#<File:"myopenfile.dat">  
Gamma> eval (read(fp));
  5  
Gamma> fp = open ("myopenfile.dat", "r", t);
#<File:"myopenfile.dat">  
Gamma> princ(read_line(fp), "\n");
(setq y 5)
  t  
Gamma> fp = open ("myopenfile.dat", "r", t);
#<File:"myopenfile.dat">  
Gamma> eval (read(fp));
Error: ./generate.slg: line 1: Malformed expression within ()  
Error: ./generate.slg: line 1: Unexpected end of file  
Macro read left extra stuff on the LISP stack: 8098478, 8098470  
nil  
nil  
Gamma>
```

The following example opens and reads a file, if it exists. If not, it prints and error message.

```
if ( (fp=open("myfile","r")) != nil )
{
  local line;
  while((line = read_line(fp)) != _eof_)
  {
    princ(line, "\n");
  }
  close(fp);
}
else
{
  princ("Error : unable to open myfile for read\n");
}
```
See Also

close, fd_open, open_string, read, read_char, read_double, read_float, read_line, read_long, read_short, read_until, seek, tell, terpri, write, writec
pipe

pipe — creates a pipe.

Synopsis

pipe ()

Arguments

none

Returns

A list of the read pipe and the write pipe, each as a file pointer.

Description

This function creates an un-named pipe.

Example

Gamma> pipel = pipe();
(#<File:"read_pipe"> #<File:"write_pipe">)
Gamma> pread = car(pipel);
#<File:"read_pipe">
Gamma> pwrite = cadr(pipel);
#<File:"write_pipe">
Gamma> write (pwrite, "This is a test");

t
Gamma> read (pread);
"This is a test"
Gamma>
princ, print, pretty_princ, pretty_print

princ, print, pretty_princ, pretty_print — write to the standard output file.

Synopsis

```lisp
princ (s_exp...)
print (s_exp...)
pretty_princ (s_exp...)
pretty_print (s_exp...)
```

Arguments

`s_exp`

Any Gamma or Lisp expression.

Returns

`t`

Description

These functions write to the standard output file, typically the screen. The `princ` and `pretty_princ` functions produce formatted output, which means that special characters are not escaped, and double quotes are not printed around character strings. Output generated by `princ` cannot be read by the Lisp reader.

`print` and `pretty_print` produce Lisp-readable output. The result of reading a printed expression using a call to `read` will generate an equal expression.

`pretty_princ` and `pretty_print` generate carriage returns and spaces with the intention of formatting the output to make long or complex Lisp expressions easier for a person to read.

Examples

```
Gamma> x = "hello";
"hello"
Gamma> print (x, "\n");
"hello"\n"t
Gamma> princ (x, "\n");
hello
Gamma> >
```
princ, print, pretty_princ, pretty_print

Gamma> class C {a; b; c;}
(defclass C nil [[[a b c]])
Gamma> princ (C);
(defclass C nil [[[a b c])
Gamma> pretty_princ (C);
(defclass C nil [[a b c]])
Gamma>
Gamma> L = list (1,2,3,4,5,list (1,2,3,4,5,list (1,2,3,4,5,list (1,2,3,4,5,list (1,2,3,4,5,list (1,2,3,4,5))))))
(1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5)))))))
Gamma> princ (L);
(1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5 (1 2 3 4 5)))))))
Gamma> pretty_princ (L);
(1 2 3 4 5
   (1 2 3 4 5
      (1 2 3 4 5
         (1 2 3 4 5
            (1 2 3 4 5
               (1 2 3 4 5
                  (1 2 3 4 5
                     (1 2 3 4 5
                        (1 2 3 4 5
                           (1 2 3 4 5
                              (1 2 3 4 5
                                 (1 2 3 4 5
                                    (1 2 3 4 5
                                       (1 2 3 4 5
                                          (1 2 3 4 5
                                             (1 2 3 4 5
                                                (1 2 3 4 5
                                                   (1 2 3 4 5
                                                      (1 2 3 4 5
                                                         (1 2 3 4 5
                                                            (1 2 3 4 5
                                                              (1 2 3 4 5
                                                                 (1 2 3 4 5
                                                                    (1 2 3 4 5
                                                                       (1 2 3 4 5
                                                                          (1 2 3 4 5
                                                                             (1 2 3 4 5
                                                                                (1 2 3 4 5
                                                                                    (1 2 3 4 5
                                                                                     (1 2 3 4 5
                                                                                        (1 2 3 4 5
                                                                                            (1 2 3 4 5
                                                                                                (1 2 3 4 5
                                                                                                    (1 2 3 4 5
                                                                                                        (1 2 3 4 5
                                                                                                            (1 2 3 4 5
                                                                                                                (1 2 3 4 5
                                                                                                                    (1 2 3 4 5
                                                                                                                        (1 2 3 4 5
                                                                                                                            (1 2 3 4 5
                                                                                                                                (1 2 3 4 5
                                                                                                                                    (1 2 3 4 5
                                                                                                                                        (1 2 3 4 5
                                                                                                                                            (1 2 3 4 5
                                                                                                                                                (1 2 3 4 5
                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        (1 2 3 4 5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (1 2 3 4 5
                                                                                     

See Also

write, writec, pretty_write
**pty, ptytio**

*pty, ptytio* — run programs in a pseudo-tty.

## Synopsis

```plaintext
pty (program, arguments...? = nil)
ptytio (termios, program, arguments...? = nil)
```

## Arguments

- **program**
  A string containing the name of the program to be executed.

- **arguments**
  A string containing any command-line arguments for the program.

- **termios**
  A termios structure.

## Returns

A list of:

```plaintext
(process_id file_for_stdin file_for_stdout file_for_stderr pty_name)
```

Where:

- **process_id**
  The process ID of the program called.

- **file_for_stdin**
  A pointer to the file used for STDIN.

- **file_for_stdout**
  A pointer to the file used for STDOUT.

- **file_for_stderr**
  A pointer to the file used for STDERR.

- **pty_name**
  The path and filename of this pseudo-tty, as a string.

## Description

These functions run programs in a pseudo-tty. A Gamma program can read from either program's standard output by issuing a `read` or `read_line` call on `file_for_stdout`. The process can be reaped using `wait`.
The `ptytio` function is the same as `pty`, but the first argument is a `termios` structure. This is useful if particular terminal characteristics are required on the `pty`. The `termios` structure is only available through the `gammatios.so` dynamic library.

**Example**

This example calls `pty` on the following test program, called `testpty.g`:

```c
#!/usr/cogent/bin/gamma
princ("Test output.

princ(cadr(argv),"\n
Here we call `pty` in interactive mode, and then read the output:

```
Gamma> ptylist = pty("testpty.g", "Argument");
(4760 #<File:"testpty.g-stdin"> #<File:"testpty.g-stdout"> #<File:"testpty.g-stderr"> "/dev/ptyp0")
Gamma> read_line(caddr(ptylist));
"This software is free for non-commercial use, and no valid commercial license"
Gamma> read_line(caddr(ptylist));
"is installed. For more information, please contact info@cogent.ca."
Gamma> read_line(caddr(ptylist));
"Test output."
Gamma> read_line(caddr(ptylist));
"Argument"
Gamma>
```
read

Reads a Lisp expression from a file.

Synopsis

read (file)

Arguments

file

A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

The next Lisp expression in the file, or _eof_.

Description

This function reads one Lisp expression from the given file. The file must have been opened before this call with the open or open_string functions. White space and newline characters are ignored during a read. If the end of file is reached during the read call, the message "Unexpected end of file" is returned. read does not evaluate the expressions it reads.

Example

The file "myreadfile.dat" contains the following:

(a b (c d e))
"A message" (+ 2 3)

Successive calls to read will produce:

Gamma> fp = open ("myreadfile.dat", "r");
#<File:"myreadfile.dat">
Gamma> read (fp);
(a b (c d e))
Gamma> read (fp);
"A message"
Gamma> read (fp);
(+ 2 3)
Gamma> read (fp);
"Unexpected end of file"
Gamma>

See Also

read_char, read_double, read_float, read_line, read_long, read_short, read_until
read_char, read_double, read_float, read_long, read_short

read_char, read_double, read_float, read_long, read_short — read the next character, double, float, long or short value in binary representation from the input file.

Synopsis

read_char (file)
read_double (file)
read_float (file)
read_long (file)
read_short (file)

Arguments

file
A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

read_char returns the decimal representation of a string of length 1 containing a single character, or -1 indicating end of file.

read_double, read_float return a floating point value, or nan indicating end of file.

read_long, read_short return an integer value, or -1 indicating end of file.

Description

These functions read the next character, double, float, long or short value in binary representation from the input file, regardless of Lisp expression syntax. This allows a programmer to read binary files constructed by other programs.

Example

The file "myfile.dat" contains the following:

ajz

Successive calls to read_char will produce:

Gamma> ft = open ("myreadcfile.dat","r");
#<File:"myreadcfile.dat">  
Gamma> read_char(ft);
97
read_char, read_double, read_float, read_long, read_short

Gamma> read_char(ft);
106
Gamma> read_char(ft);
122
Gamma> read_char(ft);
-1  Gamma>

See Also

read, read_line, read_until
read_eval_file

read_eval_file — reads a file, evaluating and counting expressions.

Synopsis

```
read_eval_file (file)
```

Arguments

*file*

A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

The number of expressions that were read and evaluated.

Description

This function reads from the current location in the file to the end, evaluating its contents as Lisp expressions and counting them.

Example

The file "myevalfile.dat" contains the following:

```
(+ 3 4) 3 + 2;
```

```
Gamma> ft = open ("myevalfile.dat", "r");
#<File:"myevalfile.dat">
Gamma> read_eval_file(ft);
4
Gamma> close(ft);
Gamma>
```

See Also

`require, load, open`
read_line

read_line — reads a single line of text.

Synopsis

```
read_line (file)
```

Arguments

`file`
A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

All characters in the file up to the first newline character, as a string. If the end of file is reached, returns "Unexpected end of file".

Description

This function reads a single line of text from the given file, up to the first newline character, regardless of Lisp syntax. This allows a programmer to deal with text files constructed by other programs.

Example

An input file contains the following:

```
Lists can be
expressed as (a b c).
```

Successive calls to `read_line` will produce:

```
Gamma> ft = open ("myreadlfile.dat", "r");
"File:"myreadlfile.dat"
Gamma> read_line(ft);
"Lists can be"
Gamma> read_line(ft);
"expressed as (a b c.)."
Gamma> read_line(ft);
"Unexpected end of file"
Gamma>
```
See Also

read, read_char, read_double, read_float, read_long, read_short, read_until
read_n_chars

read_n_chars — reads and stores characters.

Synopsis

read_n_chars (file, nchars)

Arguments

file
A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

nchars
The number of characters to read.

Returns

A buffer containing the characters read. The length of the return buffer is equal to the number of characters actually read. This function returns nil if no characters could be read.

Description

This function reads the given number of characters from the file, without any form of translation, and builds a new buffer object in which to store them. If this function reaches the end of the file before all characters are read, then the buffer will be shorter than the requested number of characters.

Example

An input file contains the following:

To be or not to be, that is the question.

Successive calls to read_n_chars will produce:

Gamma> ft = open ("myreadnfile.dat", "r");
#<File:"myreadnfile.dat">
Gamma> read_n_chars(ft,15);
#{To be or not to}
Gamma> read_n_chars(ft,18);
#{ be, that is the q}
Gamma> read_n_chars(ft,18);
#{uestion.}
Gamma> read_n_chars(ft,18);
nil
Gamma>

See Also

read_char
**read_until**

read_until — reads characters, constructing a string as it goes.

**Synopsis**

```python
def read_until(file, delimiters)
```

**Arguments**

- **file**
  - A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

- **delimiters**
  - A string of delimiter characters. "" indicates white space.

**Returns**

All characters in the file up to the first occurrence of any of delimiters characters, or "Unexpected end of file".

**Description**

This function reads characters from the input file one at a time until it reaches any of the delimiters characters, constructing a string as it goes. Successive calls continue from the point of the previous read_until. If the end of file is reached, the function returns "Unexpected end of file".

**Example**

An input file contains the following:

```
Lists can be
expressed as (a b c).
```

Successive calls to `read_until` will produce:

```python
code
Gamma> ft = open("myreadlfile.dat","r");
#<File:"myreadlfile.dat">
Gamma> read_until(ft,"(");
"Lists can be\nexpressed as "
Gamma> read_until(ft,"x");
"a b c)."
Gamma> read_until(ft,"y");
"Unexpected end of file"
```
read_until

Gamma>

See Also

read, read_char, read_double, read_float, read_line, read_long, read_short
seek

seek — sets the file position for reading or writing.

Synopsis

seek (file, offset, where)

Arguments

file
A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

offset
An integer specifying the number of characters into the file, starting from where.

where
A starting point, indicated by a number:
• 0 Beginning of the file.
• 1 Current position in the file.
• 2 End of the file.

Returns

t if successful, nil if unsuccessful.

Description

This function lets you specify a position in a file to start reading or writing.

Example

The file "myseekfile" contains the following:

Now is the time for all good men and women
to come to the aid of their world.

Gamma> msk = open("myseekfile.dat", "r",nil);
#<File:"myseekfile.dat">
Gamma> seek(msk, 5, 0);
t
Gamma> read_line(msk);
"s the time for all good men and women"
Gamma> seek(msk, 2, 1);
t
Gamma> read_line(msk);
" come to the aid of their world."
Gamma> seek(msk, -15, 2);
t
Gamma> read_line(msk);
"of their world."
Gamma> seek(msk, -3, 0);
nil
Gamma>

See Also

close, open_string, read, read_char, read_double, read_float, read_line,
read_long, read_short, read_until, tell
**ser_setup**

ser_setup — sets parameters for a serial port device.

**Synopsis**

```
ser_setup (devno, baud, bits/char, parity, stopbits, min, time)
```

**Arguments**

- **devno**
  A file ID as returned from a call to `fd_open`.
- **baud**
  A legal baud rate.
- **bits/char**
  Bits per character (6, 7 or 8).
- **parity**
  "none", "even", "odd", "mark" or "space"
- **stopbits**
  Stop bits (0, 1 or 2).
- **min**
  Default minimum number of characters for a read.
- **time**
  Default inter-character timeout for a read.

**Returns**

- t on success or nil on failure.

**Description**

This function sets the most common parameters for a serial port device, as opened by a call to `fd_open`. The function is currently only available in QNX 4.

**Example**

```
Gamma> id = fd_open("/dev/ser1",O_RDWR);
4
Gamma> ser_setup(id,9600,8,"none",1,1,0);
t
```
See Also

fd_close, fd_open
**tell**

tell — indicates file position.

**Synopsis**

tell (file)

**Arguments**

file

A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

**Returns**

Current file position, as an integer.

**Description**

This function returns the current file position as an integer representing the number of characters from the beginning of the file.

**Example**

```gamma
Gamma> msk = open("myseekfile.dat", "r",nil);
#<File:"myseekfile.dat">
Gamma> read_until(msk,"f");
"Now is the time 
Gamma> tell(msk);
17
Gamma> seek(msk, 18, 1);
t
Gamma> tell(msk);
35
Gamma> msk = open("myseekfile3.dat", "w",nil);
#<File:"myseekfile3.dat">
Gamma> write(msk,"hello");
t
Gamma> tell(msk);
7
Gamma> msk = open("myseekfile3.dat", "a",nil);
#<File:"myseekfile3.dat">
Gamma> write(msk,"goodbye");
```
t
Gamma> `tell(msk)`;
16
Gamma> 

See Also

open, open_string, read, read_char, read_double, read_float, read_line, read_long, read_short, read_until, seek
terpri

terpri — prints a newline to an open file.

Synopsis

```
terpri (file?)
```

Arguments

file

A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

t if successful, otherwise nil.

Description

This function writes a newline to the open file. Any existing contents written to a file before it was opened will be deleted when the file is opened and written to.

Example

This example writes a file with the following contents, including the newline:

```
(chars (1 2 3))
(chars (4 5 6))
```

Gamma> fw = open("mytpfile.dat", "w");
#<File:"mytpfile.dat">
Gamma> write(fw, list(#chars, list(1,2,3))); t
Gamma> terpri(fw); t
Gamma> write(fw, list(#chars, list(4,5,6))); t
Gamma> close(fw); t
Gamma> fr = open("mytpfile.dat", "r", nil);
#<File:"mytpfile.dat">
Gamma> read_line(fr);
"(chars (1 2 3))"
Gamma> read_line(fr);
"(chars (4 5 6))"
Gamma> terpri();

Gamma>
unread_char

unread_char — attempts to replace a character to a file for subsequent reading.

Synopsis

```plaintext
unread_char(file, character)
```

Arguments

- **file**
  A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

- **character**
  A single character.

Returns

- `t` if the `character` could be replaced on the `file`, otherwise `nil`.

Description

This function attempts to place the given `character` back onto the `file` so that it can be read again by subsequent calls to any of the `read` family of functions. Only one character may be replaced onto a file between calls to `read`. At least one `read` call must have been made prior to calling this function.

Example

An input file contains the following:

```
ABCDE
```

A call to `unread_char` within a succession of calls to `read_char` will produce:

```plaintext
Gamma> fr = open("myunreadfile.dat", "r", t);
#<File:"myunreadfile.dat">
Gamma> read_char(fr);
65
Gamma> read_char(fr);
66
Gamma> unread_char(fr, 'A');
t
Gamma> read_char(fr);
65
```
Gamma> `read_char(fr)`;
  67
Gamma> `read_char(fr)`;
  68
Gamma>

See Also

`read`
write, writec, pretty_write, pretty_writec

write, writec, pretty_write, pretty_writec — write an expression to a file.

Synopsis

```lisp
write (file, s_exp...)
writec (file, s_exp...)
pretty_write (file, s_exp...)
pretty_writec (file, s_exp...)
```

Arguments

file
A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

s_exp
Any Gamma or Lisp expression.

Returns

t on success, otherwise nil.

Description

Writes the given expressions to the file using the same format as print. See print for more information.

writec produces the same format as princ; pretty_write produces the same format as pretty_print; and pretty_writec produces the same format as pretty_printc. Any contents written to the file before it was opened will be deleted when any of these write functions are used.

Example

```lisp
Gamma> fw = open("mywritefile.dat", "w"); #<File:"mywritefile.dat">
Gamma> write(fw,"This is on \n one line.");
t
Gamma> writec(fw,"This finishes on \n another line.");
t
Gamma> close(fw);
t
Gamma> fr = open("mywritefile.dat", "r", nil);
```
#<File:"mywritefile.dat">
Gamma> read_line(fr);
""This is on \n one line."This finishes on "
Gamma> read_line(fr);
" another line."
Gamma>

See Also

print
write_n_chars

write_n_chars — writes characters from a buffer to a file.

Synopsis

```plaintext
write_n_chars (file, buffer, nchars)
```

Arguments

- `file`  
  A file pointer to the open destination file for the characters.
- `buffer`  
  The buffer that is the source of the characters.
- `nchars`  
  The number of characters to write.

Returns

The number of characters successfully written to the file.

Description

This function reads a given number of characters from a buffer and writes them to an open file.

Example

The following example writes the characters 'e', 'f', and 'g' to a file.

```plaintext
Gamma> fw = open("mywritencharsfile.dat", "w");
#{<File:"mywritencharsfile.dat">}
Gamma> buf = buffer (101, 102, 103, 104);
#{efgh}
Gamma> write_n_chars (fw, buf, 3);
3
Gamma>
```

See Also

`write`, `writec`, `pretty_write`
File System
absolute_path

absolute_path — returns the absolute path of the given file.

Synopsis

```c
absolute_path (filename)
```

Arguments

`filename`

The name of a disk file.

Returns

The absolute path of the file.

Description

This function returns the absolute path of the given file, with extraneous `../` constructs removed, and with the full QNX 4 node number added. The filename can be relative or absolute, on any node on the network.

Example

```
Gamma> absolute_path(".profile");
"//1/home/andrewt/.profile"
```
access

access — checks a file for various permissions.

Synopsis

access (filename, mode)

Arguments

filename
The name of a file on disk.

mode
The file mode to be tested. The legal modes are discussed below.

Returns

Zero is returned if the access mode is valid, otherwise -1 is returned and the errno is set.

Description

This function checks a file for the following permissions. Two or more permissions in bitwise OR combinations can be checked at one time.

- R_OK Test for read permission.
- W_OK Test for write permission.
- X_OK Test for execute permission.
- F_OK Test for existence of file.

The library "const/Filesys.lsp" must be required to use the constants listed above.

Example

Gamma> require_lisp("const/Filesys");
"/usr/cogent/lib/const/Filesys.lsp"
Gamma> system("touch /tmp/access_test");
0
Gamma> system("chmod a=rx /tmp/access_test");
0
Gamma> access("/tmp/access_test", R_OK|X_OK);
0
Gamma> access("/tmp/access_test", W_OK);
-1
See Also

is_busy, is_file, is_readable, is_writable, errno
basename

basename — gives the base of a filename.

Synopsis

basename (filename, suffix?)

Arguments

filename
A file name as a string, as defined by the operating system.
suffix
Any ending part of the filename to exclude.

Returns

The base of the filename. If a suffix is specified, the base of the filename without the suffix.

Example

Gamma> x = basename("/usr/george/lib/misc/myfile.dat");
"myfile.dat"
Gamma> y = basename("misc/myfile.dat", ".dat");
"myfile"
Gamma>

See Also

dirname root_path
cd

cd — changes the working directory.

Synopsis

```
cd (path)
```

Arguments

`path`

A character string which defines a directory path in the current operating system.

Returns

`t` if the operation is successful, otherwise `nil`.

Description

This function changes the current working directory for subsequent file system operations.

Example

```
Gamma> cd ("/usr/local/bin");
t
```
chars_waiting

chars_waiting — checks for characters waiting to be read on a file.

Synopsis

chars_waiting (file)

Arguments

file

A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

The number of characters waiting on the open file.

Description

This function determines whether there are any characters waiting to be read on the given file. The file may be a string file (created by open_string), in which case the number of characters which have not yet been treated by a read or similar call will be returned.

If chars_waiting is to be called on a file after it has been partially read, the file must be unbuffered first with unbuffer_file. Otherwise characters will be read in buffer by buffer and held locally in groups of 1024. This will cause chars_waiting to return unexpected results.

Example

Gamma> ft = open("mytestfile.dat", "r");
#<File:"mytestfile.dat">
Gamma> unbuffer_file(ft);
#<File:"mytestfile.dat">
Gamma> chars_waiting(ft);
9
Gamma> char(read_char(ft));
"A"
Gamma> chars_waiting(ft);
8
Gamma> read_line(ft);
"BCDEFGHI"
Gamma>
Gamma> x = open_string("hello");
#<File:"String">
Gamma> chars_waiting(x);
5
Gamma> char(read_char(x));
"h"
Gamma> chars_waiting(x);
4
Gamma>

See Also

open, open_string, unbuffer_file
directory

directory — returns the contents of a directory.

Synopsis

directory (path, filetypes, fullpaths)

Arguments

path
A path to a directory as defined by the operating system.

filetypes
A number in the range 0 to 2:
• 0 Find all files and directories.
• 1 Find all files.
• 2 Find all directories.

fullpaths
If non-nil, show the full pathname of the file by prepending the path to all filenames.

Returns

A list containing all of the requested directory entries as strings.

Example

Gamma> directory("/usr",0,nil);
("local" "lib" "bin" "readme")
Gamma> sort(directory("/usr",2,t),strcmp);
(="/usr/bin" "/usr/lib" "/usr/local")
dirname

dirname — returns the directory path of a file.

Synopsis

```
dirname (filename)
```

Arguments

*filename*

A file name as a string, including its directory path, as defined by the operating system.

Returns

The directory path of the *filename*, or if no path is entered, the *filename*.

Description

This function reads the *filename* and directory path as a string, returning the directory path as a string.

Example

```
Gamma> x = dirname("/usr/george/lib/misc/myfile.dat");
"/usr/george/lib/misc"
Gamma> y = dirname("misc/myfile.dat");
"misc"
Gamma> z = dirname("myfile.dat");
"myfile.dat"
Gamma>
```

See Also

`basename`, `root_path`
Drain — modifies end-of-file detection.

Synopsis

\[
\text{drain (file, drain_p)}
\]

Arguments

- **file**
  - An open file.
- **drain_p**
  - A flag. If non-nil, sets the file to drain.

Returns

The previous state of the drain flag for this file.

Description

This function sets a flag on the file state such that if the **drain_p** flag is on, the first time that a read on that file finds no characters waiting, the read will return immediately with "Unexpected end of file". This is intended for use in situations where the operating system may never actually generate an end of file indication, but where it is known that no more input will be available once a read would block. This function does not affect **dev_read**.

For best results, the file should be unbuffered first with **unbuffer_file**. Otherwise characters will be read in buffer by buffer and held locally in groups of 1024. This could cause a read function to return "Unexpected end of file" even when there are still characters waiting to be read.

Example

```
Gamma> fp = open("mydrainfile.dat", "r", nil);
#<File:"mydrainfile.dat">
Gamma> unbuffer_file(fp);
#<File:"mydrainfile.dat">
Gamma> drain(fp, t);
nil
Gamma> read_line(fp);
"This is my drain file."
Gamma> read_line(fp);
"Unexpected end of file"
```
Gamma>
file_date

file_date — gives the file modification date.

Synopsis

file_date (filename)

Arguments

filename

A filename as defined by the operating system.

Returns

The modification date of the file as an integer if the file exists and is readable, else nil.

Example

Gamma> fd = (file_date("myfile.dat"));
936977583
Gamma> date_of(f);d;
"Fri Sep 10 11:33:03 1999"
Gamma> file_date("nonexistent.file");
nil
Gamma> file_date("unreadable.file");
nil
Gamma>

See Also

clock, date_of
file_size

file_size — gives the file size.

Synopsis

file_size (filename)

Arguments

filename
A file name as a string, as defined by the operating system.

Returns

The size of the file in bytes if the file exists and is readable, else nil.

Example

Gamma> file_size("myfile.dat");
1467
Gamma> file_size("non_existing.file");
nil
Gamma> file_size("unreadable.file");
nil
Gamma>
flush

flush — flushes any pending output on a file or string.

Synopsis

    flush (file)

Arguments

    file

A file pointer to a previously opened file. This may be either a file in the file system, or a string opened for read and write.

Returns

    t

Description

This function flushes any pending output on the file or string. This has the effect of printing output on the screen or updating a file on disk in the case of a file. flush has no effect on strings. flush is called automatically by close.

Example

    Gamma> fp=open("myflushfile.dat","w",nil);
    #<File:"myflushfile.dat">
    Gamma> write(fp, "I am written.");
    t
    Gamma> fp=open("myflushfile.dat","r",nil);
    #<File:"myflushfile.dat">
    Gamma> read_line(fp);
    "Unexpected end of file"

    Gamma> fp=open("myflushfile.dat","w",nil);
    #<File:"myflushfile.dat">
    Gamma> write(fp, "I am written.");
    t
    Gamma> flush(fp);
    t
    Gamma> fp=open("myflushfile.dat","r",nil);
    #<File:"myflushfile.dat">
    Gamma> read_line(fp);
"I am written."
Gamma>

See Also

open, open_string
**getcwd**

getcwd — gets the current working directory.

**Synopsis**

```markdown
getcwd ()
```

**Arguments**

none

**Returns**

The current working directory as a string.

**Example**

```markdown
Gamma> getcwd();
"/home/robert/w/devel/lisp"
Gamma>
```
is_busy

is_busy — determines if a file is busy.

Synopsis

is_busy (path)

Arguments

path
A character string defining a file path and file name in this file system.

Returns

t if the named file exists and is busy, otherwise nil.

Description

This function is supported only by certain operating system and hardware combinations that mark files as busy when they are opened for write by another task. You can check this using the ls -l shell command. If it shows a busy file with a 'B' or 'b' as the first bit in the bitmask, this function should be supported.

Example

Gamma> is_busy("/tmp/busyfile");
t

See Also

is_writable
is_dir

is_dir — determines if a file is a directory.

Synopsis

is_dir (path)

Arguments

path

A character string defining a relative or absolute file path in this file system.

Returns

t if the named file exists and is a directory, otherwise nil.

Description

This function checks if a file is a directory. Relative file paths are relative to the current working directory.

Example

Gamma> is_dir("/home/robert/w/devel/lisp");
t
Gamma> is_dir("../..doc");
t
Gamma> is_dir("doc");
nil
Gamma>

See Also

is_file
is_file

is_file — determines if a file exists.

Synopsis

```plaintext
is_file (path)
```

Arguments

`path`

A character string defining a file path and file name in this file system.

Returns

`t` if the named file exists and is a regular file, otherwise `nil`.

Example

```plaintext
Gamma> is_file("/usr/doc/FAQ/txt/FAQ");
 t
 Gamma>
```

See Also

`is_dir`
is_readable

is_readable — determines if a file is readable.

Synopsis

```c
is_readable (path)
```

Arguments

```c
path
```
A character string defining a file path and file name in this file system.

Returns

`t` if the named file exists and is readable, otherwise `nil`. Existing files might not be readable because of settings on the files bitmask.

Example

```c
Gamma> is_readable("/usr/doc/FAQ/txt/FAQ");
t
Gamma>
```

See Also

`is_writable, is_busy`
**is_writable**

is_writable — determines if a file is writable.

**Synopsis**

```lisp
is_writable (path)
```

**Arguments**

*path*

A character string defining a file path and file name in this file system.

**Returns**

*t* if the named file exists and is writable, otherwise *nil*.

**Example**

```
Gamma> is_writable("/usr/doc/FAQ/txt/FAQ");
nil
Gamma> is_writable("/home/robert/w/devel/lisp/mytestfile.dat");
t
Gamma>
```

**See Also**

*is_readable*
mkdir

mkdir — creates a new sub-directory.

Synopsis

```bash
mkdir (dirname, mode)
```

Arguments

dirname
The name of the directory to create.

mode
The access permissions of the new directory, joined in sequence. If there are more than one, they are OR'ed by the | character in text format, or written consecutively in octal format. (See below.)

Returns

Zero if successful, otherwise non-zero, and the errno will be set.

Description

This function creates a new sub-directory whose path-name is `dirname`. The file permissions for the new sub-directory are determined from the `mode` argument. Valid modes are summarized here.

<table>
<thead>
<tr>
<th>Text format</th>
<th>Octal format</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_IRWXU</td>
<td>0o7</td>
<td>Read, write, execute/search</td>
</tr>
<tr>
<td>S_IRUSR</td>
<td>0o4</td>
<td>Read permission</td>
</tr>
<tr>
<td>S_IWUSR</td>
<td>0o2</td>
<td>Write permission</td>
</tr>
<tr>
<td>S_IXUSR</td>
<td>0o1</td>
<td>Execute/search permission</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text format</th>
<th>Octal format</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_IRWXG</td>
<td>0o7</td>
<td>Read, write, execute/search</td>
</tr>
<tr>
<td>S_IRGRP</td>
<td>0o4</td>
<td>Read permission</td>
</tr>
<tr>
<td>S_IWGRP</td>
<td>0o2</td>
<td>Write permission</td>
</tr>
<tr>
<td>S_IXGRP</td>
<td>0o1</td>
<td>Execute/search permission</td>
</tr>
</tbody>
</table>
Table 14. Other permission modes

<table>
<thead>
<tr>
<th>Text format</th>
<th>Octal format</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_IRWXO</td>
<td>0o7</td>
<td>Read, write, execute/search</td>
</tr>
<tr>
<td>S_IROTH</td>
<td>0o4</td>
<td>Read permission</td>
</tr>
<tr>
<td>S_IWOTH</td>
<td>0o2</td>
<td>Write permission</td>
</tr>
<tr>
<td>S_IXOTH</td>
<td>0o1</td>
<td>Execute/search permission</td>
</tr>
</tbody>
</table>

Miscellaneous permissions.

- **S_IREAD** Same as S_IRUSR
- **S_IWRITE** Same as S_IWUSR
- **S_IEXEC** Same as S_IXUSR

These flags are bitwise OR-ed together to get the desired mode.

Error constants for this function:

- **EACCES** Search permission for some component of the path denied.
- **EEXIST** The named file exists.
- **EMLINK** Maximum sub-dirs. reached.
- **ENAMETOOLONG** The name of the path or the new directory is too long.
- **ENOENT** The specified path does not exist.
- **ENOSPC** No space left on the file system.
- **ENOSYS** This function is not supported for this path.
- **ENOTDIR** A component of the passed path is not a directory.
- **EROFS** Tried to create a directory on a read-only file system.

Example

```lisp
Gamma> require_lisp("const/Filesys");
"/usr/cogent/lib/const/Filesys.lsp"
Gamma> mkdir("/tmp/mydir", S_IRWXU|S_IRGRP|S_IROTH|S_IXOTH);
0
Gamma> mkdir("/tmp/mydir2", 0o755);
0
Gamma>
```

See Also

`unlink`
path_node

path_node — gives the node number of a path in a QNX 2 path definition.

Synopsis

path_node (path)

Arguments

path
Any legal QNX 2 path.

Returns

The node number portion of the path in a QNX 2 path definition. If the path does not contain an explicit node portion, the function returns nil.

Description

This function is for version 2 of the QNX operating system only.

Example

Gamma> path_node("[2]3:/user");
2
Gamma> path_node("3:/user");
0
rename

rename — renames a file.

Synopsis

```
rename (filename, new_name)
```

Arguments

- **filename**
  - The name of a file on disk.
- **new_name**
  - The new name for the file.

Returns

- `t` if the file could be renamed, otherwise `nil`.

Description

This function makes an operating system call `rename` a file on disk. The exact behavior of this function depends on the renaming facility for the operating system.

Example

```
Gamma> rename("myfile.dat","x/myrenamedfile.dat");
t
Gamma> rename("x/myrenamedfile.dat","myfile.dat");
t
Gamma>
```
root_path

root_path — strips the final file or directory name from a path.

Synopsis

```bash
root_path (path)
```

Arguments

- `path`  
  A file path name, as a string.

Returns

The portion of the `path` with the file name and any trailing directory separators removed.

Description

This function strips the final file or directory name from the `path` to produce its parent. Any trailing directory separators are also removed. If the `path` represents the root of the file system then it is unmodified.

Example

```bash
Gamma> x = "/usr/doc/FAQ";
"/usr/doc/FAQ"
Gamma> x = root_path(x);
"/usr/doc"
Gamma> x = root_path(x);
"/usr"
Gamma> x = root_path(x);
"/
Gamma> x = root_path(x);
"/
```

See Also

- `basename`, `dirname`


tmpfile

tmpfile — generates temporary output file names.

Synopsis

tmpfile (file_prefix?)

Arguments

file_prefix
A optional string specifying the beginning of a file name.

Returns

A string representing a file name which is guaranteed not to exist at the time that the function was called.

Description

This function is used to generate a temporary output file name. The file_prefix can specify any part of a file path. If the file_prefix is nil, then "/tmp/lisp_t" will be used. Typically the resulting file name will be the result of appending a number to the file prefix.

Example

Gamma> tmpfile();
"/tmp/lisp_t1"
Gamma> tmpfile("/tmp/atempfile");
"/tmp/atempfile2"
Gamma> tmpfile("/tmp/atempfile");
"/tmp/atempfile3"
Gamma> tmpfile("/tmp/atempfile");
"/tmp/atempfile4"
Gamma> tmpfile("anotherfile");
"anotherfile5"
Gamma>
unbuffer_file

unbuffer_file — causes a file to be treated as unbuffered on both input and output.

Synopsis

unbuffer_file (file)

Arguments

file

A file pointer to a previously opened file. This may only be a file in the file system, and not a string opened for read and write.

Returns

The unbuffered file object on success, or nil on failure.

Description

This function causes a file to be treated as unbuffered on both input and output. The normal buffering mode of a file depends on whether it is in the file system, or to a character device such as a terminal or console.

When the file is unbuffered, all input and output to that file will occur immediately, without going through internal buffers. In general, an unbuffered file is much less efficient for file I/O. Unbuffering is temporary, as the file will revert to a buffered state when it is closed or reopened.

Example

Gamma> fu = open("mytestfile.dat","r",nil);
#<File:"mytestfile.dat">
Gamma> unbuffer_file(fu);
#<File:"mytestfile.dat">
Gamma>
unlink

unlink — deletes a file.

Synopsis

unlink (filename)

Arguments

filename
A string representing a valid file name.

Returns

t if the file could be deleted, otherwise nil.

Description

This function deletes a file in the file system. It will fail if the given file does not exist, or the calling process does not have sufficient privilege to delete the file. Wild cards are not expanded in the file name.

Example

Gamma> fu = open("todeletefile.dat","r");
#<File:"todeletefile.dat">
Gamma> unlink("todeletefile.dat");
t
Gamma> unlink("todeletefile.dat");
nil
Gamma>

See Also

open
OS APIs
atexit

atexit — evaluates code before exiting a program.

Synopsis

    atexit (code)

Arguments

code

    Code to be evaluated.

Returns

    The result of evaluating code.

Description

    This function gives a program an opportunity to evaluate specified code before it exits. The code should be protected from evaluation using the quote operator #.

Example

    Running this program...

    #!/usr/cogent/bin/gamma

    // Program name: exiting.g
    // Demonstrates the atexit() function.

    atexit(#princ("Exiting now.\n");
    princ("Started running...\n");
    princ("Still running.\n");
    exit_program(7);
    princ("You missed this part.\n");

    ...gives these results:

    [sh]$ exiting.g
    Started running...
    Still running.
    Exiting now.
    [sh]
block_signal, unblock_signal

block_signal, unblock_signal — delimit signal blocking.

Synopsis

block_signal (signo)
unblock_signal (signo)

Arguments

signo

The integer signal number as defined by the operating system. Symbols such as SIGINT are defined to provide an operating-system independent method for specifying this number. (see signal)

Returns

t

Description

block_signal causes a particular signal to be blocked until a call to unblock_signal is made. If the signal actually occurred while it was blocked, it will occur immediately when unblock_signal is called. Multiple occurrences of the signal while it was blocked will cause the signal to be reported multiple times when unblock_signal is called on most operating systems. Code that blocks signals should be surrounded by a call to unwind_protect.

Example

Gamma> block_signal(14);
t
Gamma> kill(getpid(),14);
t
Gamma> unblock_signal(14);
Alarm clock

Gamma> block_signal (SIGINT);
t
Gamma> critical_function();
<function return>
Gamma> unblock_signal (SIGINT);
t
See Also

block_timers, unblock_timers
**errno**

`errno` — detects and numbers errors.

**Synopsis**

```c
errno ();
```

**Arguments**

none

**Returns**

The system error number.

**Description**

When a function fails it returns a value and optionally sets the system error number. The `errno` function can be used to check the current error number. To check error numbers against constant error code in your program remember to include the file with: `require_lisp("Errno.lsp");`.

Calling the `errno` function in interactive mode does not return a valid number since you are retrieving the `errno` of the C function `printf` of the error to the screen (which will usually be 0).

**Example**

In this example, we first define a function to remove a file. Then we call that function on a non-existing file to generate an error. Finally, we check the returned error code to get the error message.

```c
function remove_file(file)
{
    unlink(file);
    errno();
}
```

Gamma> `ret_val = remove_file("/tmp/xyz");`
2
Gamma> `strerror(2);`
"No such file or directory"
Gamma>
See Also

error, strerror
exec

exec — executes a program.

Synopsis

```exec (program, arguments?...)```

Arguments

- **program**
  The name of a program to execute, as a string.

- **arguments**
  The arguments to the program, each as a string.

Returns

Does not return if successful, or -1 if an error occurs.

Description

This function is a binding for the C function `execvp`, which causes the interpreter to terminate immediately, and to run the named program in its place. The "p" in the `execvp` function indicates that a search is made for the named executable in the current path, as defined by the PATH shell variable. Unlike the C `execvp` function, the first argument in Gamma's `exec` function does not repeat the program name—it is automatically inserted for you.

Example

```
Gamma> exec("/bin/ls","-l","/usr/bin");
```

See Also

fork, wait
exit_program

exit_program — terminates the interpreter.

Synopsis

exit_program (return_value)

Arguments

return_value

An integer value to be returned to the operating system when the program exits.

Returns

This function does not return.

Description

Terminate the interpreter immediately and return the provided integer value to the operating system as an exit code.

Example

Running this program...

```
#!/usr/cogent/bin/gamma

//Program: exiting.g

atexit(#princ("Exiting now.\n"));
princ("Started running...\n");
princ("Still running.\n");
exit_program(7);
princ("You missed this part.\n");
```

...gives these results:

```
[]$ gamma exiting.g
Started running...
Still running.
Exiting now.
[]$
```

Exit showing abnormal termination of -1 (255) to the operating system.
exit_program(-1);

/>echo $?  
255
/>
fork

fork — duplicates a process.

Synopsis

fork ()

Arguments

none

Returns

A positive task id that identifies the child process, and 0 that identifies the parent process to the child; or -1 if an error occurred. The errno is set if an error occurs.

Description

The fork function creates a new process identical to the calling (parent) process except for a unique process ID. The child process has a different parent process ID and its own copy of the parent file descriptors. The child process does not inherit outstanding signals.

Example

The following example illustrates using the fork function with if syntax. This is a useful way of separating the two, identical processes produced from fork. The first block of code applies to the parent process, while the second block applies to the child.

#!/usr/cogent/bin/gamma

if ((childID = fork()) > 0)
{
    princ("P> My ID is: ", getpid(),"\n");
    princ("P> My child's ID is: ", childID, "\n");
    signal(SIGCHLD, #princ("P> Signal received that my child -- ",
                          childID, " -- has died.\n");
    princ("P> Waiting for my child.\n");
    w = wait(childID);
    princ("P> wait() returned this: ", w, "\n");
}
else
{
    sleep(2);
if (childID == -1)
    error("C> An error occurred.\n");
else
{
    princ("C> I am the child process.\nC> My process ID is: ",
         getpid(), "\n");
    sleep(2);
    princ("C> Time to exit.\n");
    exit_program(3);
}

Will produce these results:

P> My ID is: 1225
P> My child's ID is: 1226
P> Waiting for my child.

(after 2 seconds)

C> I am the child process.
C> My process ID is: 1226

(after 2 more seconds)

C> Time to exit.
P> Signal received that my child -- 1226 -- has died.
P> wait() returned this: (1226 3 nil nil)

See Also

exec, wait
**getenv**

getenv — retrieves the value of an environment variable.

**Synopsis**

```plaintext
getenv (envar)
```

**Arguments**

`envar`

A string.

**Returns**

A string containing the value of the given environment variable, or `nil` if the environment variable is not defined.

**Description**

This function retrieves the value of an environment variable from the current process's environment. The environment variable must have been set or defined previously by a call to `setenv`.

**Example**

```plaintext
Gamma> setenv("high", "40");
t
Gamma> getenv("high");
"40"
Gamma> low = 20;
20
Gamma> getenv("low");
nil
Gamma>
```

**See Also**

`setenv`
gethostname

gethostname — gets the computer's host name.

Synopsis

gethostname ()

Arguments

none

Returns

The host name of this computer, as a string.

Example

Gamma> gethostname();
"rex"
Gamma>
getnid

getnid — returns the local node number.

Synopsis

getnid ()

Arguments

none

Returns

The node number

Example

Gamma> getnid();
2

See Also

getpid
getpid

getpid — returns the program ID.

Synopsis

getpid ()

Arguments

none

Returns

The program ID of the current session of the interpreter.

Example

Gamma> getpid();
  8081
Gamma>

See Also

getnid
**getsockopt, setsockopt**

getsockopt, setsockopt — get and set a socket option.

**Synopsis**

```plaintext
getsockopt (socket, option)
setsockopt (socket, option, value1, value2? = nil)
```

**Arguments**

socket

The file descriptor of a socket.

option

The option being queried. Supported options and their possible values are listed below.

value

The value to set the socket option to. There may be one or two values, depending on the option. If a socket option requires two values, both must be specified.

**Returns**

getsockopt returns the socket option value(s) on success, as shown below, or `nil` on failure. When the option has two values, they are returned as a list.

setsockopt returns 0 on success, otherwise -1.

**Description**

These functions get and set a socket option, using the socket’s file descriptor. The supported socket options are given below.

SO_SNDTIMEO, SO_RCVTIMEO, SO_SNDLOWAT and SO_RCVLOWAT are not supported by all operating systems.

<table>
<thead>
<tr>
<th>Option</th>
<th>Possible Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO_BROADCAST</td>
<td>0 for off, non-zero for on.</td>
<td>Allows for broadcasting datagrams from the socket.</td>
</tr>
<tr>
<td>SO_DEBUG</td>
<td>0 for off, non-zero for on.</td>
<td>Records debugging information.</td>
</tr>
<tr>
<td>SO_DONTROUTE</td>
<td>0 for off, non-zero for on.</td>
<td>Sends messages directly to the network interface instead of using normal message routing.</td>
</tr>
</tbody>
</table>
### Option | Possible Values | Comments
--- | --- | ---
SO_ERROR | A number. | Resets the error status (for `getsockopt` only).
SO_KEEPALIVE | 0 for off, non-zero for on. | Transmits messages periodically on a connected socket. No response means the connection is broken.
SO_LINGER | Two values: `on_or_off` and `linger_time`, where `on_or_off` is 0 for off, non-zero for on. If on, a value for `linger_time` is required. | Keeps the socket open after a `close()` call, to deliver untransmitted messages. If `on_or_off` is non-zero, the socket will block for the duration of the `linger_time` or until all messages have been sent.
TCP_NODELAY | 0 for enable, non-zero for disable. | Disables the Nagle algorithm for sending data.
SO_OOBINLINE | 0 for off, non-zero for on. | Puts out-of-band data in the normal input queue.
SO_REUSEADDR | 0 for off, non-zero for on. | Permits the reuse of local addresses for this socket.
SO_RCVBUF | A number. | The size of the input buffer.
SO_RCVLOWAT | A number. | Sets the minimum count for input operations.
SO_RCVTIMEO | Two values: `seconds` and `nanoseconds`. | Sets a timeout value for input.
SO_SNDBUF | A number. | The size of the output buffer.
SO_SNDLOWAT | A number. | Sets the minimum count for output operations.
SO_SNDTIMEO | Two values: `seconds` and `nanoseconds`. | Sets a timeout value for output.
SO_TYPE | A number. | The type of socket (for `getsockopt` only).

### Example

```gamma
Gamma> skt = tcp_connect("localhost", 22);
8
Gamma> getsockopt(skt, SO_KEEPALIVE);
0
Gamma> setsockopt(skt, SO_KEEPALIVE, 1);
0
Gamma> getsockopt(skt, SO_DEBUG);
1
Gamma> getsockopt(skt, SO_DEBUG);
0
```
Gamma> `setsockopt(skt, SO_DEBUG, 1);`  
   -1  
Gamma> `getsockopt(skt, SO_DEBUG);`  
   0  
Gamma>
**kill**

```
kill — sends a signal to a process.
```

**Synopsis**

```
kill (pid, signo)
```

**Arguments**

- *pid*
  - The process id number.
- *signo*
  - The signal number, normally one of the built-in signal values.

**Returns**

```
t
```

**Description**

This process functions similarly to the `kill` shell command. Signals and their descriptions can be found in `signal`.

**Example**

```
Process 1:
Gamma> getpid();
8299
Gamma>
```

```
Process 2:
Gamma> kill(8299, 9);
t
Gamma>
```

```
Process 1:
Gamma> Killed
```

```
Process 3:
```
kill

Gamma> getpid();
9041
Gamma> kill (9041,14);
Alarm clock

See Also

signal
nanosleep

nanosleep — pauses the interpreter for seconds and nanoseconds.

Synopsis

nanosleep (seconds, nanosecs)

Arguments

seconds
The number of seconds to pause.

nanosecs
The number of nanoseconds to pause.

Returns

t after the time has elapsed.

Description

This function will pause the interpreter for the total time of seconds + nanoseconds.

Example

Gamma> time(1, nanosleep( 0, 999999999 ));
1.0009529590606689453
//this example is done with the ticksize at 0.5 ms.

See Also

sleep
setenv

setenv — sets an environment variable for the current process.

Synopsis

setenv (envar, value)

Arguments

envar
The name of the environment variable to set.

value
The string value for this environment variable.

Returns

t on success, or nil on failure.

Description

This function sets an environment variable for the current process. Both arguments are strings. The value of an environment variable can be acquired using the function getenv.

Example

Gamma> setenv("high", "40");
 t
Gamma> getenv("high");
"40"
Gamma> low = 20;
 20
Gamma> getenv("low");
nil
Gamma>

See Also

getenv
shm_open

shm_open — opens shared memory objects.

Synopsis

```c
shm_open (share_name, open_flags, create_mode, size?)
```

Arguments

- `share_name`
  The name of the shared memory object.
- `open_flags`
  Open control flags.
- `create_mode`
  Creation mode.
- `size`
  The size of the shared object in bytes.

Returns

A handle to the shared memory object, or `nil` on failure.

Description

This function is a wrapper for the C function `shm_open`. It is currently only available in QNX 4.

The name of the shared memory object is usually a name found under the `/dev/shmem` directory. Direct shared memory access to devices is achieved through a `shm_open` call to the existing Physical shared memory.

If you are accessing the existing Physical shared memory region (`/dev/shmem/Physical`) DO NOT use the `size` argument, as you may inadvertently resize this shared memory. The `size` argument is added as a convenience, and can be used to specify the size of a newly created object.

Valid open-flags are OR-ed combinations of:

- `O_RDONLY` Open for read-only
- `O_RDWR` Open for read and write access
- `O_CREAT` creates a new shared memory segment with access privileges governed by the `create_mode` parameter
• **O_EXCL** Exclusive mode. If O_EXCL and O_CREAT are set then shm_open will fail if the shared memory segment exists.

• **O_TRUNC** If the shared memory object exists, and it is successfully opened O_RDWR, the object is truncated to zero length and the mode and owner are unchanged.

The creation mode is usually an octal number in the range 0o000 - 0o777 defining the access privileges for the shared memory object. Require the 'const/filesys' file to load constants to make this arg easier.

Possible errno values are:

• **EACCESS** Permission to create the shared memory object denied

• **EEXIST** O_CREAT and O_EXCL are set and the named shared memory object already exists

• **EINVAL** The function call was interrupted by a signal

• **EMFILE** Too many file descriptors in use by this process

• **ENAMETOOLONG** The length of the name arg is too long

• **ENFILE** Too many shared memory objects are currently open in the system

• **ENOENT** O_CREAT is not set and the named shared memory object does not exist, or O_CREAT is set and either the name prefix does not exist or the name arg is an empty string

• **ENOSPC** Not enough space for the creation of the new shared memory object

• **ENOSYS** This function is not supported by this implementation.

**Example**

```c
// This code maps the first 1000 bytes from video memory (0xA0000) into a buffer named buf.
require_lisp("const/filesys");
require_lisp("const/mman");
fd = shm_open("Physical", O_RDONLY, 0o777);
buf = mmap(1000, PROT_READ, MAP_SHARED, fd, 0xA0000);
```

**See Also**

* shm_unlink*
shm_unlink

shm_unlink — removes shared memory objects.

Synopsis

```c
shm_unlink (share_name)
```

Arguments

`share_name`
The name of the shared object to delete.

Returns

t on success, or nil on failure, with errno set.

Description

This function is currently only available in QNX 4. It attempts to remove the shared object, `share_name`. If more than one process or link into the shared memory area exists the shared object will not be removed.

Possible values of errno are:

- **EACCESS** Permission to unlink the object is denied
- **ENAMETOOLONG** The length of the name of the object is too long
- **ENOENT** The named shared memory object does not exist.
- **ENOSYS** This function is not supported by this implementation.

Example

```c
Gamma> shm_unlink("card_mem");
Gamma>
```

See Also

`shm_open`
signal

signal — defines an expression to be evaluated at an OS generated signal.

Synopsis

```
signal (signal, action[, action]...)
```

Arguments

- `signal`
  A signal number. Normally one of the built-in signal values.
- `action`
  Any Gamma or Lisp expression.

Returns

`t`

Description

This function defines an expression to be evaluated whenever the operating system generates signal number `signal` to this process. A signal handler may be of any complexity, though it is advisable to keep signal handlers as simple as possible. All signals and timers are blocked for the duration of the signal handler. In addition, the signal handler runs in a separate, smaller heap. If the signal handler is large, this could result in memory inefficiency. Signal handlers are typically used to ensure that the Gamma application does not exit when a signal occurs.

Table 15. Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGABRT</td>
<td>Abort signal from the <code>abort()</code> C function.</td>
</tr>
<tr>
<td>SIGALRM</td>
<td>A timer has occurred. This signal is reserved in most operating system implementations of Gamma for use with the <code>after</code>, <code>at</code>, and <code>every</code> functions. This signal is not available in Linux because it is used by the timer processing internally to Gamma. In QNX, it is the timer signal from the <code>alarm()</code> C function.</td>
</tr>
<tr>
<td>SIGBUS</td>
<td>Bus error.</td>
</tr>
<tr>
<td>SIGCHLD</td>
<td>Child died. Generated when a child process of the current process has died.</td>
</tr>
<tr>
<td>SIGCONT</td>
<td>Continue. Causes the task to restart after a SIGSTP.</td>
</tr>
<tr>
<td>SIGFPE</td>
<td>Floating point exception. Generated by an illegal mathematical function call (such as division by zero).</td>
</tr>
<tr>
<td>Signal</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SIGHUP</td>
<td>Hangup. Typically generated when a terminal session disconnects.</td>
</tr>
<tr>
<td>SIGILL</td>
<td>Illegal instruction. This is an internal error.</td>
</tr>
<tr>
<td>SIGINT</td>
<td>Keyboard interrupt. Generated by <code>CTRL-C</code>.</td>
</tr>
<tr>
<td>SIGIO</td>
<td>I/O processing is required. This signal is generated when a socket or file descriptor has incoming data which must be processed.</td>
</tr>
<tr>
<td>SIGIOT</td>
<td>IOT trap. A synonym for SIGABRT.</td>
</tr>
<tr>
<td>SIGKILL</td>
<td>Killed. Kills the process with extreme prejudice. This signal cannot be caught.</td>
</tr>
<tr>
<td>SIGPIPE</td>
<td>Broken pipe. This occurs when a TCP/IP socket or a pipe to an inferior process is broken.</td>
</tr>
<tr>
<td>SIGPOLL</td>
<td>A pollable event. Synonym of SIGIO.</td>
</tr>
<tr>
<td>SIGPWR</td>
<td>Power failure. This is generated by a power monitor program to indicate that a power loss is imminent.</td>
</tr>
<tr>
<td>SQUIT</td>
<td>Quit.</td>
</tr>
<tr>
<td>SIGSEGV</td>
<td>Segmentation fault. This signal is generated by an attempt to access illegal memory. If this signal occurs, it represents a fault in the LISP interpreter and should be reported along with the corresponding memory address of the fault.</td>
</tr>
<tr>
<td>SIGSTOP</td>
<td>Stop execution immediately. This is used by the operating system to implement multi-tasking. This signal cannot be caught.</td>
</tr>
<tr>
<td>SIGSYS</td>
<td>Bad argument to a system routine. This happens very rarely.</td>
</tr>
<tr>
<td>SIGTERM</td>
<td>Terminated. This is generated by other programs which wish to terminate the job.</td>
</tr>
<tr>
<td>SIGTRAP</td>
<td>Trace/breakpoint trap.</td>
</tr>
<tr>
<td>SIGTSTP</td>
<td>Terminal stop. This signal is generated when the user attempts to stop a process (in operating systems which support job control).</td>
</tr>
<tr>
<td>SIGTIN</td>
<td>Terminal input is available.</td>
</tr>
<tr>
<td>SIGTTOU</td>
<td>Terminal output is required.</td>
</tr>
<tr>
<td>SIGURG</td>
<td>Urgent. An urgent condition has occurred.</td>
</tr>
<tr>
<td>SIGUSR1</td>
<td>User-defined signal 1.</td>
</tr>
<tr>
<td>SIGUSR2</td>
<td>User-defined signal 2.</td>
</tr>
<tr>
<td>SIGWINCH</td>
<td>Window change. This is used to indicate that a change has been made to the size or position of the window in which the process is running.</td>
</tr>
</tbody>
</table>
Example

Gamma> getpid();
10341
Gamma> signal(SIGUSR1,#princ("Got the signal.\n"));
Gamma> kill(10341,SIGUSR1);
Got the signal.

See Also

after, at, every
sleep, usleep

sleep, usleep — suspend execution.

Synopsis

sleep (seconds)
usleep (microseconds)

Arguments

seconds
The integer number of seconds to sleep.
microseconds
The integer number of microseconds to sleep.

Returns

\t

Description

These functions suspend execution for the given number of seconds or microseconds, after which time the task continues. Signals and timers will still be processed during this time.

This function is ignored in Windows, as it would cause all scripts to hang, but is maintained for compatibility. Instead, you can use a `.TimerAfter` timer, like this:

```
.TimerAfter (3, `some_code);
```

For example, to call the "myMethod" method with no arguments:

```
.TimerAfter(3, `(@self).myMethod());
```

Or, for example, to print "end":

```
.TimerAfter(3, `princ("end\n");
```

Example

Gamma> sleep (3);

(after 3 seconds...)

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
t
Gamma> **usleep (500000);**

(after 1/2 second...)

t
Gamma> 

**See Also**

nanosleep
**strerror**

*strerror* — retrieves an error message.

**Synopsis**

```c
strerror (errno)
```

**Arguments**

*errno*

The error number as returned by *errno*.

**Returns**

An error message as a string.

**Description**

This function looks up error messages associated with error numbers.

**Example**

In this example, we first define a function to remove a file. Then we call that function on a non-existing file to generate an error. Finally, we check the returned error code to get the error message.

```c
function remove_file(file)
{
    unlink(file);
    errno();
}
Gamma> ret_val = remove_file("/tmp/xyz");
2
Gamma> strerror(2);
"No such file or directory"
Gamma>
```

**See Also**

*errno*
system

system — treats its argument as a system command.

Synopsis

\[
\text{system (command\_line)}
\]

Arguments

command\_line
A string.

Returns

A numerical return code as generated by the operating system.

Description

This function treats its argument as a command to be run in the native operating system. This function will wait until the command completes before returning with the command's exit status. In UNIX and QNX 4, the command may be run in the background by using an & symbol after the command\_line argument.

Example

\begin{verbatim}
Gamma> system("ps");
   PID TTY   TIME CMD
   7856 pts/4  00:00:00 bash
   8335 pts/4  00:00:00 Gamma
   8336 pts/4  00:00:00 ps
   0
Gamma> system("ls *ty*");
li_type.c li_type.o primity.c pty.lsp
   0
Gamma> system("mysubtask &");
   0
\end{verbatim}
tcp_accept

tcp_accept — forks a new TCP socket on the server side to accept a new connection.

Synopsis

tcp_accept (socket)

Arguments

socket
A descriptor for a listening socket, as returned by a call to tcp_listen.

Returns

A file descriptor for a new, connected socket.

Description

This function allows a passive, listening socket to accept a connection, by spawning a new socket that maintains the connection. It is essentially the same as the C accept function, but returns a socket descriptor instead of an address.

Example

<table>
<thead>
<tr>
<th>CLIENT SIDE:</th>
<th>SERVER SIDE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma&gt; tcp_listen(51715); 5</td>
<td>Gamma&gt; tcp_accept(51715);</td>
</tr>
<tr>
<td>Gamma&gt; tcp_connect(&quot;localhost&quot;, 51715); 5</td>
<td>6</td>
</tr>
<tr>
<td>Gamma&gt; fd_write(5, &quot;Hi there&quot;); 8</td>
<td>Gamma&gt; fd_read(6, &quot;Hi there&quot;); 8</td>
</tr>
</tbody>
</table>

See Also

tcp_connect, tcp_listen
**tcp_connect**

tcp_connect — creates a client-side TCP socket connection.

**Synopsis**

```c
tcp_connect (host, port)
```

**Arguments**

- **host**
  - The IP address of the host machine.

- **port**
  - The port to connect to.

**Returns**

A file descriptor for a new, connected socket.

**Description**

This function creates a connected socket. This can be accessed with Gamma `fd_*` functions such as `fd_write` and `fd_read`.

**Example**

See the example for `tcp_accept`.

**See Also**

tcpAccept, tcpListen
**tcp_listen**

*tcp_listen* — creates a server-side TCP socket connection.

**Synopsis**

```
tcp_listen (port, backlog?)
```

**Arguments**

- **port**
  The port to connect to.

- **backlog**
  The IP address of the host machine.

**Returns**

A file descriptor for a new, connected socket.

**Description**

This function creates a connected socket. This can be accessed with Gamma `fd_*` functions such as `fd_write` and `fd_read`.

**Example**

See the example for `tcp_accept`.

**See Also**

`tcp_accept`, `tcp_connect`
wait

wait — waits for process exit status.

Synopsis

wait (taskid?, options?)

Arguments

taskid

A process ID number. A value of 0 indicates any process.

options

Wait option WNOHANG or WUNTRACED.

Returns

One of three possibilities:

- A list of four items:
  1. The process ID.
  2. The process exit status (WEXITSTATUS), or nil.
  3. A termination signal (WTERMSIG) if the process exited due to a signal, or nil.
  4. A stopped signal (WSTOPSIG) if the process stopped due to a signal, or nil.
- t if the WNOHANG option had been set and there was a child with a status change.
- nil if there was a failure due to error.

Description

This function combines and simplifies the C functions wait and waitpid in a single function. If taskid is provided, then the function acts as waitpid, and will not return until the given child task has died.

The WNOHANG option allows the calling process to continue if the status of specified child process is not immediately available. The WUNTRACED option allows the calling process to return if the child process has stopped and its status has not been reported. Both of these can be specified using the OR (| |) operator.

Example

Process 1:

Gamma> child = fork();
9089
Gamma> 0
Gamma> if (child > 0) wait(); else exit_program(5);

Process 2:

 Gamma> kill(9089,14);
        t
 Gamma>

Process 1:

 (9089 nil 14 nil)
 Gamma>

See Also

fork, exec
Dynamic Loading
AutoLoad

AutoLoad — allows for run-time symbol lookup.

Synopsis

AutoLoad ("pattern", `action)

Arguments

pattern
A shell style pattern.

action
An action to be taken when the pattern is matched.

Returns

The _auto_load_alist_, which is a list of all currently stored AutoLoad rules. Each rule is itself formatted as a list. This is the _auto_load_alist_ syntax:

`((pattern action_func action_arg ...) ...)`

The members of each rule list are as follows:

pattern
The AutoLoad pattern parameter.

action_func
The function specified in the AutoLoad action parameter.

action_arg
The function argument(s) specified in the AutoLoad action parameter.

For example, the AutoLoad rules in AutoLoadLib.g (at the time of this writing) would be returned as follows:

`(("P[Tthg]*" DllLoad "libgammaph.so") ("gl[A-Z]*" DllLoad "libgammagl.so")
("GLUT_*" DllLoad "libgammagl.so") ("GLU_*" DllLoad "libgammagl.so")
("GL_*" DllLoad "libgammagl.so") ("ASCII_*" DllLoad "libgammagl.so")
("KB_*" DllLoad "libgammagl.so") ("GM_*" DllLoad "libgammamgl.so")
("EVT_*" DllLoad "libgammamgl.so") ("[mM][gG][lL]*" DllLoad "libgammamgl.so")
("[gG]tk*" DllLoad "gammagtk.so"))`

Description

This function gives Gamma a way to look up symbols during run-time. If Gamma comes across an undefined symbol while executing a program, and if the symbol matches the
pattern, then Gamma executes the action. Normally the action is either a direct definition of the symbol, or an attempt to load a DLL that defines the symbol, using DllLoad, for example.

The available patterns are as follows:

- * matches any number of characters, including zero.
- [c] matches a single character which is a member of the set contained within the square brackets.
- [\c] matches any single character which is not a member of the set contained within the square brackets.
- ? matches a single character.
- {xx,yy} matches either of the simple strings contained within the braces.
- \c (a backslash followed by a character) - matches that character.

This function is not part of the base Gamma executable. It is provided by a Gamma library AutoLoadLib.g which can be accessed using the Gamma require function like this:

```
require ("/usr/cogent/require/AutoLoadLib.g");
```

**Example**

- In this example, we use the ClearAutoLoad function to clear the AutoLoad list just to make the steps easier to follow.
- Once a library is loaded or a symbol is defined, Gamma no longer sends a "Looking for symbol" message.
- Notice how although NoAutoLoad and ClearAutoLoad remove a pattern from future consideration, any symbols defined or any libraries loaded before they were called remain valid.

```gamma
Gamma> require ("/usr/cogent/require/AutoLoadLib.g");
Gamma> ClearAutoLoad();
nil
Gamma> AutoLoad ("[gG]tk*", `DllLoad ("gammagtk.so"));
(("[gG]tk*" DllLoad "gammagtk.so"))
Gamma> gtk_arg_new;
Looking for gtk_arg_new
(defun gtk_arg_new (arg_type) ...)
Gamma> gtk_main;
(defun gtk_main () ...)
Gamma> testvar;
Looking for testvar
Symbol is undefined: testvar
```
debug 1> (Ctrl - D)
Gamma> AutoLoad("testvar", 'testvar = 5);
(("testvar" setq testvar 5) ([gG]tk* DllLoad "gammagt.k.so"))
Gamma> testvar;
Looking for testvar
5
Gamma> NoAutoLoad("testvar");
(("[gG]tk*" DllLoad "gammagt.k.so"))
Gamma> testvar;
5
Gamma> ClearAutoLoad();
nil
Gamma> gtk_main;
(defun gtk_main () ...)
Gamma> gtk_false;
(defun gtk_false () ...)
Gamma>

See Also

ClearAutoLoad, NoAutoLoad, DllLoad
autoload_undefined_symbol

autoload_undefined_symbol — checks undefined symbols for AutoLoad.

Synopsis

```
autoload_undefined_symbol (!sym)
```

Arguments

`sym`
A symbol.

Returns

```
nil on success, else error.
```

Description

This function is generally used internally by the AutoLoadLib.g program. It is the default function that is called when an undefined symbol is encountered at run-time, if the AutoLoad.g library has been required into the program. This is normally done by startup.g, which is automatically loaded by the Gamma executable at startup.

Example

In this example, the first symbol (`test1`) is checked by autoload_undefined_symbol from within the AutoLoadLib.g program. We know this because the message "Looking for symbol" indicates that Gamma had to use AutoLoadLib.g to get the definition of the symbol. For the second symbol (`test2`), we make the autoload_undefined_symbol call ourselves, and the "Looking for symbol" doesn't appear. This indicates that Gamma knew the value of the symbol and didn't have to use AutoLoadLib.g to look it up.

```
Gamma> AutoLoad("test1", `test1 = 9); 
(("test1" setq test1 9) ("P[Tthg]*" DllLoad "libgammaph.so")...)
Gamma> test1;
Looking for test1
9
Gamma> AutoLoad("test2", `test2 = 8);
(("test2" setq test2 8) ("test1" setq test1 9) 
 ("P[Tthg]*" DllLoad "libgammaph.so")...)
Gamma> autoload_undefined_symbol(test2);
nil
Gamma> test2;
8
Gamma>
```
See Also

AutoLoad
AutoMapFunction

**AutoMapFunction** — maps a C function to a Gamma function.

**Synopsis**

```
AutoMapFunction (name, rettype, args)
```

**Arguments**

- `name`
  The name of a C function.
- `rettype`
  Not yet documented.
- `args`
  Not yet documented.

**Returns**

Not yet documented.

**Description**

This function checks to see if the C function `name` exists in (is linked into) the Gamma executable. If so, it then maps it to a Gamma function according to the `rettype` and `args`. The details this function have not yet been documented.

**See Also**

- `AutoLoad`
ClearAutoLoad

ClearAutoLoad — removes all AutoLoad rules.

Synopsis

```
ClearAutoLoad ()
```

Arguments

None.

Returns

`nil`.

Description

This function removes all `AutoLoad` rules by setting the `_auto_load-alist_` to `nil`.

This function is not part of the base Gamma executable. It is provided by a Gamma library `AutoLoadLib.g` which can be accessed using the Gamma `require` function like this:

```
require ("/usr/cogent/require/AutoLoadLib.g");
```

Example

See the example for `AutoLoad`.

See Also

`AutoLoad, NoAutoLoad`
dlclose

dlclose — closes an open dynamic library.

Synopsis

dlclose (handle)

Arguments

handle

The "handle" returned by dlopen.

Returns

0 when successful, else -1.

Description

This function is a wrapper for the dlclose shell command. Each call decrements the link count in the dl library created by dlopen. When this count reaches zero and no other loaded libraries use symbols in it, the library is unloaded.

If the library exports a routine named _fini, that will be called just before the library is unloaded.

Example

Gamma> dlopen("libform.so",RTLD_NOW|RTLD_GLOBAL);
134940024
Gamma> a = dlopen("libform.so",RTLD_NOW|RTLD_GLOBAL);
134940024
Gamma> dlclose(a);
0
Gamma> dlclose(a);
0
Gamma> dlclose(a);
-1
Gamma>

See Also

dlopen
dlerror

dlerror — reports errors in dl functions.

Synopsis

dlerror ()

Arguments

none

Returns

An error message, or 0 if no error has occurred since it was last called.

Description

This function returns an error message for the most recent error in dlopen or dlclose. If several errors have occurred since the last call to dlerror, only the first will return an error message.

Example

Gamma> dlopen("nolibraryhere",RTLD_LAZY);
0
Gamma> dlopen("norhere",RTLD_LAZY);
0
Gamma> dlerror();
"norhere: cannot open shared object file: No such file or directory"
Gamma> dlerror();
nil
Gamma>

See Also

dlopen, dlclose
dlfunc

dlfunc — reserved for future use.

Synopsis

dlfunc (handle symname rettype args)

Arguments

Returns

Description

Example

dlfunc (handle symname rettype args)

See Also
DllLoad

DllLoad — loads dynamic libraries.

Synopsis

```scheme
DllLoad ("filename", verbose? = nil)
```

Arguments

- `filename`:
  The name of the dynamic library to be loaded.
- `verbose`:
  When set to `t`, shows the paths of load attempts.

Returns

An integer "handle" on success, or an error message.

Description

This function loads a DLL if the system supports it (Linux, QNX 6, and MS-Windows). The first search path for the DLL is taken to be `.`, next is `/usr/cogent/dll/`, and finally the system DLL search path, if any.

Example

Without using the optional `verbose` parameter:

```scheme
Gamma> DllLoad("gammagtk.so");
135024608
Gamma>
```

Using the optional `verbose` parameter:

```scheme
Gamma> DllLoad("gammagtk.so", t);
DllLoad: attempting to load: ./gammagtk.so
DllLoad: attempting to load: /usr/cogent/dll/gammagtk.so
135024608
Gamma>
```

See Also

`AutoLoad`
dlmethod

*dlmethod* — reserved for future use.

**Synopsis**

```
dlmethod (handle class methodname symname rettype args)
```

**Arguments**

**Returns**

**Description**

**Example**

**See Also**
NoAutoLoad

NoAutoLoad — removes selected AutoLoad rules.

Synopsis

NoAutoLoad ("pattern")

Arguments

pattern

A shell style pattern.

Returns

The _auto_load alist_ (a list of all currently stored AutoLoad rules) with the rules corresponding to the pattern removed.

Description

This function removes from future consideration any AutoLoad rules that correspond to the pattern.

The available patterns are as follows:

- * matches any number of characters, including zero.
- [c] matches a single character which is a member of the set contained within the square brackets.
- [^c] matches any single character which is not a member of the set contained within the square brackets.
- ? matches a single character.
- {xx, yy} matches either of the simple strings contained within the braces.
- \c (a backslash followed by a character) - matches that character.

This function is not part of the base Gamma executable. It is provided by a Gamma library AutoLoadLib.g which can be accessed using the Gamma require function like this:

```
require ("/usr/cogent/require/AutoLoadLib.g");
```

Example

See the example for AutoLoad.
See Also

AutoLoad, ClearAutoLoad
dlopen

dlopen — loads a dynamic library from a file.

Synopsis

dlopen (filename flags)

Arguments

filename

The name of the file to open, as a string. If no absolute path is given, the file is searched for in the user's LD_LIBRARY path, the /etc/ld.so.cache list of libraries, and the /usr/lib/ directory.

flags

Must be either RTLD_LAZY or RTLD_NOW, optionally OR'ed with RTLD_GLOBAL.

• RTLD_LAZY causes undefined symbols to be resolved as the dynamic library code executes.

• RTLD_NOW forces undefined symbols to be resolved before dlopen returns, otherwise dlopen fails.

• RTLD_GLOBAL makes any external symbols defined in the library available to subsequently loaded libraries.

Returns

An integer "handle" if successful, else 0.

Description

This function is a wrapper for the dlopen shell command. It loads a dynamic library from the file and returns a "handle", which is an integer uniquely associated with the file for this application. The same handle is returned each time the same library is opened, and the dl library counts the number of links created for each handle.

If the library exports a routine named _init, that will be executed before dlopen returns.

Example

Gamma> dlopen("libform.so",RTLD_LAZY|RTLD_GLOBAL);
134936808
Gamma> dlopen("libconsole.so",RTLD_NOW);
0
Gamma> dlopen("libconsole.so",RTLD_LAZY);
Gamma> dlopen("libconsole.so",RTLD_LAZY);
134935848

See Also

dlclose
Profiling and Debugging
allocated_cells

allocated_cells — gives the number of allocated and free cells.

Synopsis

allocated_cells ()

Arguments

none

Returns

A list containing the number of allocated cells and the number of free cells currently held by the memory management system.

Description

The memory management system allocates cells as required to continue execution, limited only by operating system memory. Once cells have been allocated, they are placed on the heap by the garbage collector and re-used. New cells are only allocated from the operating system if the garbage collector is unable to fulfill a request for more memory from the running Gamma or Lisp program. This function returns the number of cells which are currently in use, and the number of free cells remaining on the heap. The sum of these numbers is the total number of cells allocated by the interpreter.

Example

This example shows 380 cells in use and 1620 cells free on the heap, for a total of 2000 cells available.

Gamma> allocated_cells();
(380 1620 0 0 0 0 0 0)

See Also

free_cells, gc
eval_count

eval_count — counts evaluations made since a program started.

Synopsis

eval_count ()

Arguments

none

Returns

A list of three values. First is the number of times any symbol has been evaluated. Second is the number of times any function has been evaluated. Third is the number of times any other Gamma expression has been evaluated.

Description

This function counts the number of evaluations of symbols, functions, and other Gamma expressions. All of these are counted from the time the program started.

Example

Gamma> gc();
1
Gamma> eval_count();
(0 2 0)
Gamma> a = 5;
5
Gamma> eval_count();
(0 4 1)
Gamma> a;
5
Gamma> eval_count();
(1 5 1)
Gamma>
free_cells

free_cells — returns the number of available memory cells.

Synopsis

free_cells ()

Arguments

none

Returns

The number of free memory cells available on the memory heap.

Example

Gamma> free_cells();
1620
Gamma>

See Also

allocated_cells
function_calls

function_calls — tells how often a function was called during profiling.

Synopsis

function_calls (function)

Arguments

function

A function.

Returns

The number of times this function has been called while profiling was active.

Description

This function queries the system to determine the number of times that a function was called while profiling was active (using the profile function).

Example

Gamma> profile(t);
t
for(i=0;i<10;i++)
{
   princ("i: ",i,"\n");
}
>> i:0
>> i:1
>> i:2
>> i:3
>> i:4
>> i:5
>> i:6
>> i:7
>> i:8
>> i:9
Gamma> i;
9
Gamma> profile(nil);
t
function_calls

Gamma> function_calls(princ);
10

See Also

profile, function_runtime
**function_runtime**

`function_runtime` — gives the time a function has run during profiling.

**Synopsis**

```plaintext
function_runtime (function)
```

**Arguments**

`function`

A function.

**Returns**

The total number of seconds that the `function` has run.

**Description**

This function returns the number of seconds (as a floating point number) that a function has run during all complete invocations of the function while profiling has been active. The number of seconds is measured using the QNX 4 tick clock, and thus represents elapsed time rather than CPU time, with a granularity of one tick (typically 10ms). Invocations of the function which have not completed at the time of the call to `function_runtime` are not included in the calculation.

**Example**

```
Gamma> function_runtime(cdr);
0.05341
```

**See Also**

`function_calls`, `profile`
**gc**

`gc` — runs the garbage collector.

**Synopsis**

```
gc ()
```

**Arguments**

none

**Returns**

The number of cells freed by the garbage collector.

**Description**

Causes the garbage collector to run if possible. The garbage collector will not run during a timer or signal handler, but it will flag the need for garbage collection, causing the garbage collector to run immediately after the timer or signal handler exits.

**Example**

```
Gamma> gc();
68
Gamma> gc();
17
Gamma> fp = open("myfile.dat", "r", nil);
#<File:"myfile.dat">    
Gamma> close(fp);
Gamma> gc();
67
Gamma> gc();
17
Gamma>
```

**See Also**

`allocated_cells, free_cells`
**gc_blocksize**

*gc_blocksize* — for internal use only.

**Synopsis**

```
gc_blocksize (ncells)
```
**gc_enable**

*gc_enable* — for internal use only.

**Synopsis**

```
 gc_enable (enable_p)
```
gc_newblock

gc_newblock — for internal use only.

Synopsis

gc_newblock ()
**gc_trace**

*gc_trace* — controls the tracing of garbage collection.

**Synopsis**

```
gc_trace (on_flag)
```

**Arguments**

*on_flag*

If non-`nil`, turn on garbage collector tracing, else turn it off.

**Returns**

The new status of garbage collector tracing.

**Description**

This function turns on (on-flag is non-*nil*) or off (on-flag is *nil*) the tracing of garbage collection. When garbage collection tracing is on, statistics are collected concerning the number of allocated cells, number of collection calls, and the elapsed time spent within the garbage collector. These statistics can be accessed using a call to allocated-cells.

**Example**

```
Gamma> gc_trace (t);
nil
```

**See Also**

*allocated_cells*
profile

profile — collects statistics on function usage and run time.

Synopsis

```
profile (on_p, tick_nanosecs?)
```

Arguments

```
on_p
  If non-nil, start profiling, else stop profiling.
```

```
tick_nanosecs
  Reset the QNX 4 tick size to this many nanoseconds before beginning to profile.
```

Returns

The previous state of profiling.

Description

This function starts (or stops) collecting statistics on the usage and run time of all functions in the system. The profile mechanism uses an interrupt on the QNX 4 tick clock, and so must run with root permissions. If the optional `tick_nanosecs` argument is provided, this function will reset the tick size. Otherwise, it will profile using the current tick size. The smaller the tick size, the more precise is the profile result.

Example

The following program gives the output shown below.

```
#!/usr/cogent/bin/gamma

require_lisp("Profile.lsp");

e_list = list();
j = 0;

function print_reverse()
{
  with i in cdr(argv) do
  {
    e_list = cons(i, e_list);
  }
```
j++;  
}  
princ("The numbers in reverse order are:\n", e_list, "\n");
}

function main()
{
    profile(t);
    print_reverse();
    profile(nil);
    
    profiled_functions();
    princ("Function calls: ", function_calls(cons),"\n");
    princ("Function runtime: ", function_runtime(cons),"\n");
}

Entered on command line:

[sh]$ ex_profile.g 1 2 3 4 5

Output:

The numbers in reverse order are:
(5 4 3 2 1)

<table>
<thead>
<tr>
<th>Function</th>
<th>Calls</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+++</td>
<td>5</td>
<td>4e-06</td>
</tr>
<tr>
<td>cdr</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cons</td>
<td>5</td>
<td>2e-06</td>
</tr>
<tr>
<td>for</td>
<td>1</td>
<td>2.3e-05</td>
</tr>
<tr>
<td>profile</td>
<td>1</td>
<td>1e-06</td>
</tr>
<tr>
<td>setq</td>
<td>5</td>
<td>3e-06</td>
</tr>
<tr>
<td>princ</td>
<td>1</td>
<td>0.006502</td>
</tr>
<tr>
<td>print_reverse</td>
<td>1</td>
<td>0.006534</td>
</tr>
<tr>
<td>progn</td>
<td>6</td>
<td>0.006546</td>
</tr>
</tbody>
</table>

Function calls: 5
Function runtime: 1.9999999999999999095e-06

See Also

function_calls, function_runtime
set_autotrace

set_autotrace — is reserved for future use.

Synopsis

set_autotrace (state, functions...)

Arguments

state

functions

Returns

Description

Example

See Also
set_breakpoint

set_breakpoint — is reserved for future use.

Synopsis

```c
set_breakpoint (state, functions...)
```

Arguments

- `state`
- `functions`

Returns

Description

Example

See Also
time

time — gives command execution times.

Synopsis

time (iterations, !command)

Arguments

iterations
   The number of times to execute the command.

command
   Any Gamma or Lisp command.

Returns

The number of seconds consumed performing the command for the given number of iterations.

Description

This function performs the command for the given number of iterations and returns the clock time consumed. This does not break down the time into user and system time. Times on successive calls to this function will differ slightly due to operating system requirements, garbage collection and active timers.

Example

Gamma> time(10, list(1,2,3));
3.297225339338183403e_05
Gamma>
trace, notrace

trace, notrace — turn tracing on or off.

Synopsis

```
trace (!code?)
notrace (!code?)
```

Arguments

code
   If provided, limits the scope to this code.

Returns

With no argument, t, or with a code argument, the result of evaluating code.

Description

These functions turn tracing (execution tracking to standard output) on or off, either at
the global level, or for the duration of the evaluation of code, if provided.

Example

```
#!/usr/local/bin/gamma -d

/* here is an example of a troublesome function and its
   return being debugged with the aid of trace() and
   notrace() functions.
*/

a = 0;
b = 1;
trace();
function trouble_function(x,y) {y/x;}
results = trouble_function(a,b);
notrace();
princ(results, "\n");
```

Gamma generates the following:

```
(defun trouble_function (x y) (/ y x))
--> trouble_function
```
(trouble_function a b)
  (/ y x)
  --> inf
  --> inf
(setq results (trouble_function a b))
  --> inf
(notrace)
inf
apropos

apropos — finds all defined symbols in the current interpreter environment.

Synopsis

apropos (pattern, predicate?)

Arguments

pattern
  A character string which specifies a search pattern

predicate
  A function taking one argument which will return either nil or non-nil.

Returns

A list of all symbols defined in the system which match the given pattern, and if the predicate is supplied, whose values are true under that predicate.

Description

This function searches the names of all defined symbols in the currently running interpreter environment. The pattern can contain the following special characters:

• * matches any number of characters, including zero.
• [c] matches a single character which is a member of the set contained within the square brackets.
• [^c] matches any single character which is not a member of the set contained within the square brackets.
• ? matches a single character.
• {xx,yy} matches either of the simple strings contained within the braces.
• \ (a backslash followed by a character) - matches that character.

The predicate is any function which accepts a single argument. If the predicate evaluates to non-nil when given the value of a symbol, and if the symbol matches the pattern, then the symbol will be reported by apropos. If the predicate is not supplied, then all symbols which match the pattern will be reported. The pattern is case-sensitive.

Example

Gamma> apropos("s*", function_p);
(setq strchr string symbol)
Gamma> apropos("?[sli]igc,er]*");
(SIGCHLD SIGCONT dlerror)
create_state, enter_state, exit_state

create_state, enter_state, exit_state — are part of the SCADALisp exception-driven state machine mechanism.

Synopsis

create_state (state_function, symbol?...)  
enter_state (state_machine, state)  
exit_state (state_machine, state)

Arguments

state_function
The function to call upon entering this state.

symbol
One or more symbols which will act as triggers to cause this state to be re-evaluated.

state_machine
A state machine created through a call to (new StateMachine)

state
A state created through a call to (create-state...)

Returns

create_state: The new state definition.

enter_state: A status value.

exit_state: A status value.

Description

These functions are part of the exception-driven state machine mechanism built into SCADALisp. This mechanism is not fully supported, and will not be documented for this release. The reader may find the library file StateMachine.lsp helpful in determining how to use state machines. In general, this function should not be called directly from user code, as it is designed to provide support for the StateMachine library functions.

Example

none
gensym

gensym — generates a unique symbol.

Synopsis

gensym (prefix_string?)

Arguments

prefix_string
    A character string which will be used as the prefix for the newly generated symbol.

Returns

A unique symbol.

Description

This function generates a symbol which does not currently exist by attaching a unique
number to the end of the prefix_string. If the prefix_string is nil, use a default
prefix.

Example

Gamma> gensym("tag");
tag1
Gamma> gensym();
tmp_sym2
Gamma> tag3 = 1;
1
Gamma> gensym("tag");
tag4
Gamma>
modules

modules — is obsolete, and returns nothing of value.

Synopsis

modules ()

Arguments

none
nil

Returns

A string.

Description

This function is obsolete, and returns nothing of value.
stack

stack — lists all functions called so far.

Synopsis

```
stack ()
```

Arguments

```
none
```

Returns

A list of all of the functions called up to this point in the execution of the Gamma program.

Description

A function that calls `stack` is presented in order, with the most recently called function at the end of the function list. `stack` can be useful for debugging programs by requesting a stack trace when an error occurs.

Example

The following program:

```
#!/usr/cogent/bin/gamma

function hms_to_sec(hms)
{
    hms = list_to_array(string_split(hms, ":", -1));
    (number(hms[0]) * 60 + number(hms[1])) * 60 + number(hms[2]);
    stk = stack();
}

tocheck = list(12,5,13);
hms_to_sec("tocheck");
princ(stk,"\n");
```

Yields these results:

```
((hms_to_sec tocheck) (progn (setq hms (list_to_array (string_split hms : (neg 1)))) (+ (* (+ (* (number (aref hms 0)) 60) (number (aref hms 1)) 60) (number (aref hms 2))) (setq stk #0=(stack))) #0#)
```
See Also

print_stack
add_hook

add_hook — hooks a function to an event.

Synopsis

add_hook (hook_sym, function_sym)

Arguments

hook_sym
One of several symbols used to identify the hook, as listed below.

function_sym
The function that is to run when the event occurs.

Returns

The hooked function (function_sym) that was added.

Description

This function sets up a hook, which is a function that is called when a particular event takes place. The arguments to the function identified by the function_sym are determined by the particular event. A hook function must be defined with the correct number of arguments, or else with optional or variable length arguments. The currently available hooks and the respective events that trigger their functions are as follows:

- taskstarted_hook: triggered whenever a task starts.
- taskdied_hook: triggered whenever a task dies.
- exception_hook: triggered whenever an exception for any point is emitted by the Cogent DataHub.
- echo_hook: triggered whenever an echo for any point is emitted by the Cogent DataHub.
- gc_hook: triggered whenever the garbage collector runs.
- The following are related to tracing code executions, but haven't been fully documented.

  trace_symbol_hook
  trace_entry_hook
  trace_exit_hook
  breakpoint_hook

One common use of this function is to add the internal taskstarted or taskdied functions, with the taskstarted_hook or taskdied_hook. Whenever a task that is
registered with \texttt{nserve} starts or dies, \texttt{nserve} sends a message to all Gamma applications running IPC. Any of these applications that has added the \texttt{taskstarted\_hook} or \texttt{taskdied\_hook} then runs their corresponding \texttt{function\_sym} function.

\section*{Example}

This example program requires \texttt{qserve} and \texttt{nserve} to be running. It gives the output shown below:

```
#!/usr/cogent/bin/gamma

//Program: ex_addrunhooks.g

function main ()
{
  init_ipc ("x","x");

  add_hook (#taskstarted\_hook, #hook\_started);
  add_hook (#taskdied\_hook, #hook\_died);

  run_hooks (#taskstarted\_hook, "testing start");
  run_hooks (#taskdied\_hook, "testing died");

  while(t)
    next_event();
}

function hook\_started (!a?...=nil)
{
  princ ("Hooked task started: ", a, "\n");
}

function hook\_died (!a?...=nil)
{
  princ ("Hooked task died: ", a, "\n");
}
```

\textbf{Output from} \texttt{ex_addrunhooks.g} \textbf{at startup:}

\begin{itemize}
  \item Hooked task started: (testing start)
  \item Hooked task died: (testing died)
\end{itemize}

Starting a new Gamma task named \texttt{mytask}...

```
Gamma> \texttt{init\_ipc("mytask", "myqueue")};
\texttt{t}
Gamma>
```
...elicits this output from `ex_addrunhooks.g`:

Hooked task started: (mytask default myqueue 0 0 1874 0)

Checking process status with `nsnames`:

```
[home/robert]$ nsnames
Name   Domain  Queue    NID PID
mytask default myqueue  0   1874
x      default x        0   1873
```

Terminating `mytask` elicits this output from `ex_addrunhooks.g`:

Locked task died: (mytask default myqueue 0 0 1874 0)

See Also

`run_hooks`, `remove_hook`, `init_ipc`
close_task

close_task — closes a task opened by locate_task.

Synopsis

```plaintext
close_task (task)
```

Arguments

task

A task descriptor as assigned to a locate_task call.

Returns

t if the task could be closed, else nil.

Description

When a task is opened (located) for interprocess communication, a communication link may be established. This link must be cleaned up if it is to be re-used. There is no hard limit to the number of tasks which may be open with QNX 4 message passing, but TCP/IP exerts an operating system-dependent limit on the number of simultaneously open tasks. Tasks will automatically be closed by the garbage collector when they are no longer referenced.

Example

```plaintext
Gamma> task = locate_task("Task 1",nil);
#<Task:9684>
Gamma> close_task(task);
t
Gamma>
```

See Also

locate_task
_destroy_task

_destroy_task — should never be used.

Synopsis

This function should not be used under any circumstances.
init_async_ipc

init_async_ipc — requests queue information from a task.

Synopsis

```plaintext
init_async_ipc (other_task)
```

Arguments

other_task

A task descriptor as assigned to a locate_task call.

Returns

Non-nil on success, or nil on failure.

Description

This function initializes the interprocess communication system to allow this task to make calls to register_point, register_existing_point, send_async and send_string_async. It requests queue information from the given task. A deadlock situation could occur if two tasks attempt to initialize asynchronous communication with one another at the same time. The queue server task, `qserve`, must be running for this call to succeed.

Example

```
Gamma> init_ipc("mytask","mytask_q");
t
Gamma> task = locate_task("server", t);
#<Task: 32271>
Gamma> init_async_ipc(task);
t
```

See Also

init_ipc, locate_task, register_point, send_async, send_string_async
**init_ipc**

*init_ipc* — sets up necessary data structures for IPC.

**Synopsis**

```
init_ipc (my_name, my_queue_name?, domain?)
```

**Arguments**

- **my_name**
  
  A name for this task, as a string. It is only used internally.

- **my_queue_name**
  
  Optional queue name for this task, as a string. This is necessary for asynchronous communication, and it must be unique on the system.

- **domain**
  
  Optional domain name for this task.

**Returns**

- t on success, otherwise nil.

**Description**

Sets up all of the data structures needed prior to attempting any interprocess communication from this task. Messages can be neither sent nor received before this call is made. All Cogent DataHub functions use IPC. If the value of `my_queue_name` is `nil`, no queue name is assigned and no asynchronous IPC is possible.

**Example**

```
Gamma> init_ipc("myname","myqueue");
t
Gamma>
```

**See Also**

- isend, next_event, next_event_nb, read_point, read_existing_point, register_point, send, send_async, send_string, send_string_async, write_point, write_existing_point
isend

isend — sends a synchronous message and doesn't wait for the result.

Synopsis

isend (task, s_exp)

Arguments

task
A task descriptor as assigned to a locate_task call.

s_exp
Any Gamma or Lisp expression.

Returns

t if the message was sent successfully, otherwise nil.

Description

This function sends a message via synchronous interprocess communication, but does not wait for the result. The receiving task must respond immediately, prior to actually evaluating the message that was sent. The result code can only show whether the message was sent successfully. This is a compromise between synchronous and asynchronous messaging techniques.

Example

Gamma> init_ipc("mytask","myqueue");
t
Gamma> task = locate_task("other_task",nil);
<task id>
Gamma> isend(task,#list(do_something));
t
See Also

locate_task, send, send_async
locate_task

locate_task — finds and connects to tasks by name.

Synopsis

locate_task (task_name, async_reqd)

Arguments

task_name
  The name of the task to locate. The other task must have declared this name through
  init_ipc or name_attach.
async_reqd
  t if locate_task should automatically call init_async_ipc for this task.

Returns

A task if successful, otherwise nil.

Description

This function makes a call to the name locator task for the current operating system. If it
finds the named task it makes an IPC connection (in TCP/IP) or creates a virtual circuit (in
QNX 4) to that task. If async_reqd is t, then init_async_ipc is also called.

When Gamma locates a task, it returns a printed representation of it, which
looks like this: #<Task:10120>. This representation cannot be read back into
Gamma, so a symbol is usually assigned when calling locate_task to facili-
tate referring to or working with a task. We refer to this symbol as the task de-
scriptor. For instance, in the example below, the symbol tsk is the task de-
scriptor.

Example

The two tasks are initiated with init_ipc before calling locate_task.

Task 1:

Gamma> init_ipc("first","Q1");
t
Gamma> getpid();
tsk = locate_task("second", nil);
#<Task:9092>
send (tsk, #princ("Are you there?\n"));

Task 2:

init_ipc("second","Q2");
getpid();
next_event
Are you there?

See Also

locate_task_id
locate_task_id

locate_task_id — finds and connects to tasks by task ID and network node.

Synopsis

locate_task_id (task_id, node_id, channel_id, async_reqd)

Arguments

task_id
The task ID for this task (as a number).

node_id
The network node number for this task.

channel_id
The task ID for this task. This is required for QNX 6, ignored in QNX 4 and Linux.

async_reqd
  t if locate_task should automatically call init_async_ipc for this task.

Returns

A task if successful, otherwise nil.

Description

This function makes a TCP/IP connection or QNX 4 virtual circuit to the named task based on the task_id and the node number on which the task is running. If async_reqd is t, then init_async_ipc is also called. If the node number is zero, the current node is used.

Example

Task 1:

Gamma> init_ipc("Task 1","14");
  t
Gamma> getpid();
  9271
Gamma>

Task 2:

Gamma> init_ipc("Task 2","25");
  t
Gamma> `locate_task_id(9271,1,nil);`
#<Task:9271>
Gamma>

See Also

`locate_task`
name_attach

name_attach — attaches a name to a task.

Synopsis

name_attach (task_name)

Arguments

task_name
The name to attach to this task.

Returns

t if the name was successfully attached, otherwise nil.

Description

This function sends a message to the QNX 4 name locator task (nameloc) to attach a name on this node. If the name locator is not running or the name has already been attached by another task, the call will fail.

Example

// attach my name
Gamma> name_attach("firstname");
t
// attach an alternate name
Gamma> name_attach("pseudonym");
t
// attempt to attach my name again
Gamma> name_attach("firstname");
nil
**nserve_query**

*nserve_query* — puts information from *nserve* into an array.

**Synopsis**

```c
nserve_query()
```

**Arguments**

none

**Returns**

An array of instances of the class *TaskInfo*, or *nil* on failure.

**Description**

This function retrieves all the information available in the Cascade NameServer (*qserve*), and puts it into an array. Each item in the array is an instance of the *TaskInfo* class, as returned from the function *task_info*. Please refer to the documentation of that function for more details.

This function requires that *init_ipc* be called first.

**Example**

```c
Gamma> init_ipc("a", "aq");
Gamma> pretty_princ(nserve_query(), "\n");
[(TaskInfo (channel_id . 0) (domain . toolsdemo) (name . /dh/toolsdemo)
  (node_id . 0) (node_name . 0) (pid . 5394)
  (queue_name . /dh/toolsdemo) (queue_size . 0))
(TaskInfo (channel_id . 0) (domain . toolsdemo) (name . control)
  (node_id . 0) (node_name . 0) (pid . 16995)
  (queue_name . controlq) (queue_size . 0))
(TaskInfo (channel_id . 0) (domain . toolsdemo) (name . emul)
  (node_id . 0) (node_name . 0) (pid . 16998)
  (queue_name . emulq) (queue_size . 0))
(TaskInfo (channel_id . 0) (domain . default) (name . a)
  (node_id . 0) (node_name . 0) (pid . 16999)
  (queue_name . aq) (queue_size . 0))]
```
See Also

task_info
remove_hook

remove_hook — removes a hooked function.

Synopsis

remove_hook (hook_sym, function_sym)

Arguments

hook_sym
One of several symbols used to identify a hook, as listed below.

function_sym
The function that is to be removed.

Returns

The hooked function (function_sym) that was removed.

Description

This function removes a hook that was previously set up with add_hook. The currently available hooks are:

taskstarted_hook
taskdied_hook
exception_hook
echo_hook
gc_hook
trace_symbol_hook
trace_entry_hook
trace_exit_hook
breakpoint_hook

Example

Modifying the example in add_hook by adding one line:

... 
add_hook (#taskstarted_hook, #hook_started);
add_hook (#taskdied_hook, #hook_died);
run_hooks (#taskstarted_hook, "testing start");
run_hooks (#taskdied_hook, "testing died");
/* Remove the hook */
remove_hook (#taskstarted_hook, #hook_started);

while(t)
...

would remove the taskstarted_hook.

See Also

add_hook, run_hooks, init_ipc
run_hooks

run_hooks — runs a hooked function.

Synopsis

run_hooks (hook_sym, args\ldots?)

Arguments

hook_sym
One of several symbols used to identify a hook, as listed below.

args
The arguments of the function that is to run when the event occurs.

Returns

t on success or \texttt{nil} on failure.

Description

This function runs a hook that was previously set up with \texttt{add_hook}. The currently available hooks are:

\begin{verbatim}
  taskstarted_hook
taskdied_hook
exception_hook
echo_hook
gc_hook
trace_symbol_hook
trace_entry_hook
trace_exit_hook
breakpoint_hook
\end{verbatim}

Example

Please refer to the example in \texttt{add_hook}.

See Also

\texttt{remove\_hook, init\_ipc}
**send**

send — transmits expressions for evaluation.

**Synopsis**

```
send (task, s_exp)
```

**Arguments**

- **task**
  A task descriptor as assigned to a `locate_task` call.
- **s_exp**
  Any Gamma or Lisp expression.

**Returns**

A result depending on the receiving task, which could include:

- `t` if the message was delivered successfully.
- `nil` if the message could not be delivered.
- An expression in the form: `(error "error message")` if there was an error. See `error`.

**Description**

This function constructs an ASCII string representing the `s_exp` and transmits it via synchronous interprocess communication to the receiving `task`. The `task` processes the message and returns a result based on that processing. If the `task` is another Gamma process, the message will be interpreted as a Gamma expression and evaluated. The return value will be the result of that evaluation.

**Example**

Task 1:

```
Gamma> init_ipc ("a","a");
t
Gamma> tsk = locate_task("b",nil);
<Task:9751>
Gamma> send(tsk, #princ("hello
"));
hello
t
Gamma> send_async(tsk, #princ(cos(5), "\n"));
t
```
send(tsk, #princ("goodbye\n"));
t
Gamma> 

Task 2:

Gamma> init_ipc ("b","b");
t
Gamma> while(t) next_event();
hello
0.28366218546322624627
goodbye

See Also

isend, locate_task, send_async, send_string, send_string_async
send_async

send_async — transmits expressions asynchronously.

Synopsis

```
send_async (task, s_exp)
```

Arguments

- **task**: A task descriptor as assigned to a `locate_task` call.
- **s_exp**: Any Gamma or Lisp expression.

Returns

- `t` if the message was successfully delivered, otherwise `nil`.

Description

This function constructs a string representation of the given expression and delivers it via asynchronous interprocess communication to the receiving task. If the message could not be delivered, `send_async` returns `nil`. There is no indication of the status of the receiving task as a result of processing the message.

Example

**Task 1:**

```
Gamma> init_ipc ("a","a");
t
Gamma> tsk = locate_task("b",t);
#<Task:9751>
Gamma> send_async(tsk, #princ("hello, b\n"));
t
Gamma> send_async(tsk, #princ(cos(5), "\n"));
t
Gamma>
```

**Task 2:**

```
Gamma> init_ipc ("b","b");
t
```
Gamma> while(t) next_event();
hello, b
0.28366218546322624627

See Also

isend, locate_task, send, send_string, send_string_async
send_string

send_string — transmits strings for evaluation.

Synopsis

send_string (task, string)

Arguments

task
A task descriptor as assigned to a locate_task call.

string
Any string.

Returns

A result depending on the receiving task.

Description

This function transmits the string via synchronous interprocess communication to a non-Cogent DataHub receiving task. The task processes the message and returns a result based on that processing. If the task is a Gamma process, the message will be interpreted as a Lisp expression and evaluated. The return value will be the result of that evaluation. If an error occurs during the evaluation, an expression of the form: (error "error message") will be returned. If the message could not be delivered, nil is returned.

Example

Gamma> a = 5;
5
Gamma> b = 6;
5
Gamma> send_string(task, string("(+",a," ",b," )"));
11

See Also

isend, locate_task, send, send_async, send_string_async
**send_string_async**

Send_string_async — transmits a string asynchronously.

**Synopsis**

```plaintext
send_string_async (task, string)
```

**Arguments**

- `task`
  A task descriptor as assigned to a `locate_task` call.

- `string`
  A string.

**Returns**

- `t` if the message was successfully delivered, otherwise `nil`.

**Description**

This function delivers the `string` via asynchronous interprocess communication to a non-Cogent DataHub receiving `task`. If the message could not be delivered, `send_string_async` returns `nil`. There is no indication of the status of the receiving `task` as a result of processing the message.

**Example**

Task 1:

```plaintext
Gamma> init_ipc ("a","a");
t
Gamma> tsk = locate_task("b",t);
#<Task:9751>
Gamma> send_string_async(tsk, "2 + 2");
t
Gamma> send_string_async(tsk,string(list(#a,#b,#c)));
t
Gamma>
```

Task 2:

```plaintext
Gamma> init_ipc ("b","b");
t
```

Gamma> while(t) next_event();

See Also

isend, locate_task, send, send_async, send_string
taskdied, taskstarted

**taskdied, taskstarted** — internal functions that call another function when a task starts or stops.

**Synopsis**

```plaintext
 taskdied (task_name, qname, domain, node, task_id)
 taskstarted (task_name, qname, domain, node, task_id)
```

**Arguments**

- `task_name`  
  The name of the task which started or stopped.

- `node`  
  The node on which the task started or stopped.

- `task_id`  
  The process ID for the task.

- `qname`  
  The name of the task's queue, if any.

- `domain`  
  The Cogent DataHub domain for this task.

**Returns**

User-defined.

**Description**

These functions are internal to Gamma. They call `run_hooks (#taskstarted_hook, args...)` and `run_hooks (#taskdied_hook, args...)` respectively. They are called whenever a task registered with the Cascade NameServer (**ns**erver) starts or stops. You can set up hooks to use these functions through the `add_hook` function.

These functions were originally available to programmers, and have been internalized to allow for the greater flexibility of the `add_hook` and `run_hook` functions. However, if you have existing code that you don’t want to change, you can define your own versions of `taskdied` and `taskstarted` that shadow the built-in functions and do what they always used to do. Your old code will not break, but it will hide the hook version of the `taskdied` and `taskstarted` functions.

On the other hand, you could get both with something like this:
builtin_taskdied = taskdied;
builtin_taskstarted = taskstarted;

function main ()
{
    init_ipc ("x","x");

    add_hook (#taskdied_hook, #hook_taskdied);
    add_hook (#taskstarted_hook, #hook_taskstarted);

    while(t)
        next_event();
}

function taskdied (!a?...=nil)
{
    princ ("task died: ", a, "\n");
    funcall (builtin_taskdied, a);
}

function taskstarted (!a?...=nil)
{
    princ ("task started: ", a, "\n");
    funcall (builtin_taskstarted, a);
}

function hook_taskdied (!a?...=nil)
{
    princ ("hook task died: ", a, "\n");
}

function hook_taskstarted (!a?...=nil)
{
    princ ("hook task started: ", a, "\n");
}
task_info

task_info — gets information from a task descriptor.

Synopsis

task_info (tsk)

Arguments

 tsk
     A task descriptor, as returned by the locate_task function.

Returns

 An instance of the class TaskInfo, or nil on failure.

Description

This function returns an instance of Gamma's TaskInfo class. The instance variables of this class correspond to information contained in the task descriptor, as follows:

  channel_id
     The channel ID number, which is used in QNX 6 but not in QNX 4 or Linux.
  domain
     The name of the Cogent DataHub domain for the tsk.
  name
     The name of the tsk, as recorded in the Cascade NameServer. This attribute is not contained in a task descriptor, and thus is always returned as nil from this function.
  node_id
     The node ID number.
  node_name
     The node_id expressed as a string.
  pid
     The process ID number.
  queue_name
     The name of the Cascade QueueServer queue, as registered with the Cascade NameServer.
  queue_size
     The size of the Cascade QueueServer queue.
Example

Gamma> init_ipc("a", "aq");

Gamma> tsk = locate_task("/dh/toolsdemo", nil);

#>Task:5394<

Gamma> task_info(tsk);

{TaskInfo (channel_id . 0) (domain . "toolsdemo") (name)
  (node_id . 0) (node_name . "0") (pid . 5394)
  (queue_name . "/dh/toolsde") (queue_size . 0)}

Gamma>

See Also

nserve_query
Events and Callbacks
add_set_function

add_set_function — sets an expression to be evaluated when a given symbol changes value.

Synopsis

add_set_function (symbol, s_exp)

Arguments

symbol
  A symbol.

s_exp
  Any Gamma or Lisp expression.

Returns

t

Description

This function binds an expression to be evaluated whenever the value of the symbol changes. This expression is available globally, so if the value of the symbol changes during a change in scope, the expression will be evaluated. All changes in that sub-scope will trigger new evaluations of the expression. This can be used to automatically maintain consistency between the program and a the Cogent DataHub, or to implement forward chaining in calculation rules. The expression will not be evaluated if the new value is eq to the previous value.

When a set expression (the s_exp) is being evaluated the special variables this, value and previous are all bound:

• this The symbol whose value has changed.
• value The current value of this as a result of the change.
• previous The value of this immediately prior to the change.

Example

Gamma> b = 5;
5
Gamma> add_set_function(#b,#princ("changed\n");
(princ "changed\n")
Gamma> b = 4;
The following code automatically sounds an alarm whenever a computed_tank_level rises above 10000 and silences the alarm whenever it drops below 10000. The Cogent DataHub is automatically updated to maintain the same value as the Gamma task.

```javascript
function send_to_datahub (point)
{
    write_point(point, value);
}

function check_alarm (value)
{
    if (value == 1)
        sound_alarm();
    else
        silence_alarm();

    send_to_datahub(this);
}

function check_tank_level (depth)
{
    if (depth > 10000)
        high_alarm = 1;
    else
        high_alarm = 0;

    send_to_datahub(this);
}

add_set_function(#high_alarm, #check_alarm(high_alarm));
add_set_function(#computed_tank_level,#check_tank_level(computed_tank_level));
```

See Also

when_set_fns, remove_set_function
flush_events

flush_events — handles all pending events, then exits.

Synopsis

```c
flush_events()
```

Arguments

none

Returns

The result of executing all pending events, then exits.

Description

This function ensures that an appropriate event-handling function is called to handle all pending events from: a window system (where applicable), other tasks (interprocess communication messages), timers, or signals. Upon completion, `flush_events` causes the program to exit.

Example

```
Gamma> flush_events();

(the result of any pending events)
[/user/cogent/bin]$
```

See Also

`next_event`
next_event, next_event_nb

next_event, next_event_nb — wait for an event and call the event handling function.

Synopsis

```c
next_event ()
next_event_nb ()
```

Arguments

`none`

Returns

The result of executing the next event. If no event was processed, `next_event_nb` will return undefined, and `next_event` will not return.

Description

`next_event` blocks, waiting for an event from: a window system (where applicable), another task (an interprocess communication message), a timer, or a signal. An event handling function is automatically called if one has been defined for the event. The result of `next_event` is the result returned from the event handler, or `nil` if no event handler had been defined.

`next_event_nb` behaves exactly like `next_event`, except that `next_event_nb` (nb stands for non-blocking) returns immediately with undefined if no event is waiting to be processed.

Example

1. Here is the simplest use of `next_event`, causing Gamma to wait for and process the next event.

   ```gamma
   Gamma> while(t) next_event();
   ```

2. This program does basically the same thing, creating a main loop for program event processing, but it features error protection as well.

   ```c
   while (t)
   {
     try
     {
   ```
next_event();
}
catch
{
    princ("last error: ", _last_error_," calling stack: ",
            stack(),"\n");
}

remove_set_function

remove_set_function — removes a set function from a symbol.

Synopsis

remove_set_function (symbol, s_exp)

Arguments

symbol
   The symbol from which to remove the expression.

s_exp
   An expression set for the symbol, such as that added by add_set_function.

Returns

The expression, in Lisp syntax, which was removed, or nil if none was removed.

Description

This function removes a set expression from the symbol. The s_exp is compared to all of the current expression set for the symbol using the comparison function eq.

Example

Gamma> b = 5;
5
Gamma> add_set_function(#b,#princ("changed\n"));
(princ "changed\n")
Gamma> b = 4;
changed 4
Gamma> remove_set_function(#b,#princ("changed\n"));
(princ "changed\n")
Gamma> b = 3;
3
Gamma>

See Also

add_set_function, when_set_fns
when_set_fns

when_set_fns — returns all functions set for a symbol.

Synopsis

```
when_set_fns (symbol)
```

Arguments

symbol
A symbol.

Returns

The expressions, in Lisp syntax, that have been set to be evaluated whenever the symbol's value changes.

Example

```
Gamma> b = 5;
5
Gamma> add_set_function(#b,#princ("Changed.\n");
(princ "Changed."
)
Gamma> add_set_function(#b,#princ("Update now.\n");
(princ "Update now."
)
Gamma> b = 4;
Update now.
Changed.
4
Gamma> when_set_fns(#b);
((princ "Update now."
) (princ "Changed."
))
Gamma>
```

See Also

add_set_function, remove_set_function
**after**

*a timer that initiates an action after a period of time.*

**Synopsis**

```lisp
after (seconds, action...)
```

**Arguments**

*seconds*

A number of seconds, which may be fractional.

*action*

One or more statements to be executed. This argument is evaluated, so literal statements must be quoted.

**Returns**

An integer timer number which may be used as the argument to `cancel`.

**Description**

This function specifies an action to be performed after a given period of time in seconds has elapsed. The number of *seconds* may be specified to arbitrary precision, but will be limited in fact by the timer resolution of the operating system. In most cases this is practically limited to 20 milliseconds (0.05 seconds).

The timer functions `after`, `every` and `at` all cause an action to occur at the specified time, regardless of what is happening at that time, except if the timer expires during garbage collection. In this case, the timer will be serviced as soon as the garbage collection finishes.

For Gamma to notice a timer, you must make a call to `next_event`.

**Example**

```lisp
Gamma> after(30, #princ("Time's up!\n"));
1
Gamma> next_event();
(30 seconds pass)
Time's up!
nil
```
See Also

at, every, cancel, _timers_ in Predefined Symbols
at

at — a timer that initiates an action at a given time, or regularly.

Synopsis

\texttt{at (day, month, year, hour, minute, second, actions...)}

Arguments

data
Restriction on the day of the month (1-31), or \texttt{nil} for none.

\textit{month}
Restriction on the month of the year (1-12), or \texttt{nil} for none.

\textit{year}
Restriction on the year (1994-2026), or \texttt{nil} for none.

\textit{hour}
Restriction on the hour of the day (0-23), or \texttt{nil} for none.

\textit{minute}
Restriction on the minute in the hour (0-59), or \texttt{nil} for none.

\textit{second}
Restriction on the second in the minute (0-59), or \texttt{nil} for none.

\textit{actions}
The actions to perform when the specified time arrives.

Returns

An integer number which may be used as the argument to \texttt{cancel}.

Description

This function specifies an action to be performed at a given time, or to occur regularly at certain times of the \textit{minute}, \textit{hour}, \textit{day}, \textit{month} or \textit{year}. A restriction on a particular attribute of the time will cause \texttt{at} to fire only if that restriction is true.

A restriction may be any number in the legal range of that attribute, or a list of numbers in that range. Illegal values for the time will be normalized. For example, a time specified as July 0, 1994 00:00:00 will be treated as June 30, 1994 00:00:00. If \texttt{nil} is specified for any attribute of the time, this implies no restriction and \texttt{at} will fire cyclically at every legal value for that attribute.
For Gamma to notice a timer, you must make a call to `next_event`. To notice repeating timers, the call to `next_event` can be used with a call to `while(t)`. 

**Example**

```plaintext
//To print "hello" at 12:00 noon on June 2, 1994:
at(2,6,1994,12,0,0,#princ("hello\n"));

//To print "hello" at 12:00 noon on the first day
//of every month in 1994:
at(1,nil,1994,12,0,0,#princ("hello\n"));

//To print "hello" every half minute at 30 seconds
//and on the minute on the 1st and 15th of every month
//except July and August, for any year:
at(list(1,15), list(1,2,3,4,5,6,9,10,11,12),list(0,30), #(princ "hello\n"));

//To print "hello" every 10 seconds during the hour
//of 3:00pm every December 21st.
at(21,12,nil,15,nil,list(0,10,20,30,40,50), #princ("hello\n"));
```

**See Also**

`after`, `every`, `_timers_` in *Predefined Symbols*
**block_timers, unblock_timers**

block_timers, unblock_timers — block and unblock timer firing.

**Synopsis**

```plaintext
block_timers ()
unblock_timers ()
```

**Arguments**

none

**Returns**

true

**Description**

Timers are potentially handled by a different mechanism from operating system signals. It may be desirable to block all timers from firing for the duration of an operation, which may not be possible using the `block_signal` mechanism. If a timer fires while timers are blocked, the timer function will be called as soon as timers are unblocked. This will not delay subsequent timers.

For example, if a timer is intended to fire every 5 seconds at 5, 10, ... seconds after the minute and the 5-second timer is blocked until second 7, the next timer in the sequence will still fire at 10 seconds. If the 5-second timer were blocked until second 27, then the 5, 10, 15, 20 and 25-second timers would all fire at second 27 and the next timer would fire at second 30. Code which blocks timers should be surrounded by a call to `unwind_protect`.

**Example**

```gamma
Gamma> block_timers();
t
Gamma> protected_function();
<function return>
Gamma> unblock_timers();
t
```

**See Also**

`block_signal, unblock_signal, after, at, every, _timers_ in Predefined Symbols`
cancel

_cancel_ — removes a timer from the set of pending timers.

Synopsis

```plaintext
cancel (timer_number)
```

Arguments

_timer_number_

An integer number returned from a call to `after`, `at` or `every`.

Returns

The complete timer definition for the canceled timer, or `nil` if no timer was canceled.

Description

Removes a timer from the set of pending timers based on its unique timer ID as returned by the function which created the timer. If no timer could be found with the corresponding timer number, nothing happens.

Example

To set a timer to repeat every 5 seconds, then stop it:

```lisp
 Gamma> every(5, #princ("hello\n"));  
 1  
 Gamma> cancel(1);  
 [945884155 683256506 5 ((princ "hello\n") 1]  
 Gamma>
```

See Also

_ _timers_ in Predefined Symbols

clock, nanoclock

clock, nanoclock — get the OS time.

Synopsis

clock ()
nanoclock ()

Arguments

none

Returns

The current clock value in seconds from the operating system as a long integer. nanoclock includes the nanoseconds as well.

Description

This function gets the operating system clock setting in seconds. The time is usually expressed as the number of seconds from midnight January 1, 1970 on UNIX systems, though it may differ across implementations.

Example

Gamma> clock();
999810273
Gamma> nanoclock();
999810273.66378700733
Gamma>

See Also

date, date_of
date

date — gets the OS date and time; translates seconds into dates.

Synopsis

date (seconds?, is_utc?)

Arguments

seconds
A number of seconds, such as returned from a call to clock.

is_utc
A value of t puts the date in Coordinated Universal Time (formerly known as Greenwich Mean Time, GMT).

Returns

The date as a character string.

Description

This function returns a character string which represents the current date and time in human-readable form. This form depends on the operating system, but will look like "Sat Mar 21 15:58:27 2000" on most UNIX systems. The seconds parameter returns the date that corresponds to the number of seconds since Jan 1, 1970, Coordinated Universal Time.

Example

Gamma> date();
"Fri Mar 31 09:18:27 2000"
Gamma> date(987654321);
"Thu Apr 19 00:25:21 2001"
Gamma> date(987654321,t);
"Thu Apr 19 04:25:21 2001"
Gamma> date(0,t);
"Thu Jan 1 00:00:00 1970"
Gamma>

See Also

clock
**date_of**

*date_of* — is obsolete, see *date*

**Synopsis**

```
date_of (seconds)
```

**Arguments**

*seconds*

A system time as a long integer, which may be obtained from the *clock* function.

**Returns**

The date as a character string.

**Description**

This function has been superceded by *date*. It returns a character string which represents the given date and time in human-readable form. This form depends on the operating system, but will look like "Fri Feb 16 21:50:32 1973" on most UNIX systems.

**Example**

```
Gamma> date_of(987654321);
"Thu Apr 19 00:25:21 2001"
```

**See Also**

*clock, date*
every

every — a timer that initiates an action every number of seconds.

Synopsis

every (seconds, action...)

Arguments

seconds
The number of seconds. This may be fractional. Realistically the operating system will not be able to keep up with numbers below about 0.05. This will differ from machine to machine.

action
The actions to perform continuously every given number of seconds.

Returns

An integer number which may be used as the argument to cancel.

Description

This function specifies an action to be performed every time the number of seconds elapses. The return value is a unique timer number which may be used to cancel the action prior to the time expiring by calling cancel. The number of seconds may be specified to arbitrary precision, but will be limited in fact by the timer resolution of the operating system. In most cases this is practically limited to 20 milliseconds (0.05 seconds).

The timer functions after, every and at all cause the action to occur at the specified time, regardless of what is happening at that time, except if the timer expires during garbage collection. In this case, the timer will be serviced as soon as the garbage collection finishes.

For Gamma to notice a timer, you must make a call to next_event. To notice repeating timers, the call to next_event can be used with a call to while(t).

Example

Print hello every 5 seconds.

Gamma> every(5, #princ("Hello\n"));
1
Gamma> while(t) next_event();
Hello
Hello
Hello
...

See Also

after, at, _timers_ in Predefined Symbols
gmtime

gmtime — transforms Unix time to UTC time and date in ASCII format.

Synopsis

```c
gmtime (time_t)
```

Arguments

`time_t`

The time, usually expressed as the number of seconds from midnight January 1, 1970 on UNIX systems, though it may differ across implementations.

Returns

A instance of the class `tm`, whose members are as follows:

- `.sec`
  The number of seconds after the minute (0 - 59).
- `.min`
  The number of minutes after the hour (0 - 59).
- `.hour`
  The number of hours past midnight (0 - 23).
- `.mday`
  The day of the month (1 - 31).
- `.mon`
  The number of months since January (0 - 11).
- `.year`
  The number of years since 1900.
- `.wday`
  The number of days since Sunday (0 - 6).
- `.yday`
  The number of days since January 1 (0 - 365)
- `.isdst`
  1 if daylight saving time is in effect, 0 if not, and a negative number if the information is not available.

Example

```c
Gamma> pretty_princ("UTC breakout:\t", gmtime(1149261975.5000002), "\n");
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
See Also

gmtime, mktime
localtime

localtime — transforms Unix time to local time and date in ASCII format.

Synopsis

```c
localtime (time_t)
```

Arguments

* `time_t`  
  The time, usually expressed as the number of seconds from midnight January 1, 1970 on UNIX systems, though it may differ across implementations.

Returns

An instance of the class `tm`, whose members are as follows:

- `.sec`  
  The number of seconds after the minute (0 - 59).
- `.min`  
  The number of minutes after the hour (0 - 59).
- `.hour`  
  The number of hours past midnight (0 - 23).
- `.mday`  
  The day of the month (1 - 31).
- `.mon`  
  The number of months since January (0 - 11)
- `.year`  
  The number of years since 1900.
- `.wday`  
  The number of days since Sunday (0 - 6).
- `.yday`  
  The number of days since January 1 (0 - 365)
- `.isdst`  
  1 if daylight saving time is in effect, 0 if not, and a negative number if the information is not available.

Example

```c
Gamma> pretty_princ ("Local breakout:\t", localtime (1149261975.5000002), "\n");
```

Skkynet Cloud Systems, Inc • 2233 Argentia Road • Suite 306 • Mississauga • ON • L5N 2X7 • 1.905.702.7851 • https://skkynet.com
Local breakout: (tm (hour . 11) (isdst . 1) (mday . 2) (min . 26)
   (mon . 5) (sec . 15) (wday . 5) (yday . 152) (year . 106))
t
Gamma>

See Also

gmtime, mktime
**mktime**

`mktime` — converts the ASCII date and time data in a `tm` class to Unix time.

**Synopsis**

```c
mktime (time_t)
```

**Arguments**

`tm`

A `tm` class, as created by `localtime` or `gmtime`

**Returns**

The time, usually expressed as the number of seconds from midnight January 1, 1970 on UNIX systems, though it may differ across implementations.

**Example**

```c
Gamma> princ ("Local breakout to Unix:\t", mktime (localtime(1149261975.5000002)),
          Local breakout to Unix: 1149261975
Gamma>
```

**See Also**

`gmtime`, `localtime`
**timer_is_proxy**

`timer_is_proxy` — controls timer handling in Gamma.

**Synopsis**

```
timer_is_proxy (t_or_nil);
```

**Arguments**

`t_or_nil`

An expression that evaluates to `t` or `nil`.

**Returns**

The passed argument (`t` or `nil`).

**Description**

This function controls how timers are fundamentally handled within Gamma. By default, timers are handled by the processing of proxies which allows Gamma to delay the timer, if necessary, if a critical system process is occurring.

Calling `timer_is_proxy` with `nil` makes all timers operate by using signals. In the QNX 4 operating system SIGUSER1 (SIGALRM?) is used, and the attach code is run as a handled signal.

Running timers via signals has some very dramatic consequences. When running in this mode **ALL TIMER CODE MUST BE SIGNAL SAFE.**

**Example**

```
Gamma> timer_is_proxy (nil);
nil
Gamma> timer_is_proxy (t);
 t
Gamma>
```

**See Also**

`every, at, after, block_timers, unblock_timers, cancel`
Cogent DataHub
add_exception_function, add_echo_function

add_exception_function, add_echo_function — assign functions for exceptions or echoes on a point.

Synopsis

add_exception_function (symbol, s_exp);
add_echo_function (symbol, s_exp);

Arguments

symbol
  A point name, as a symbol.

s_exp
  Any Gamma or Lisp expression.

Returns

t

Description

When a Gamma or Lisp program is run in conjunction with the Cogent DataHub, process points may change at any time, causing point change events to occur. A point change event is referred to as an exception. It is possible to bind any Gamma or Lisp expression to a symbol to be evaluated when an exception occurs. If a program can both write a point on the DataHub and react to exceptions on that point, it is possible that the DataHub will "echo" a point written by the program itself. If this is not handled, an infinite loop between the program and the DataHub could occur. The DataHub tags point echoes so that a different function can be called in the program when that echo arrives back at its origin. Only the originating task will see a point exception as an echo. All other tasks will see a normal exception.

When an exception handler (the s_exp argument) is being evaluated the special variables this, value and previous are all bound:

• this The symbol which received the exception.
• value The current value of this as a result of the exception.
• previous The value of this immediately prior to the exception.

Example

Gamma> add_exception_function(#temp, #princ("temp change\n"));
(princ "temp change\n")
Gamma> add_echo_function(#temp,nil);
nil
Gamma> next_event();
temp change
(t)
Gamma> read_point(#temp);
30
Gamma> write_point(#temp,25);
t
Gamma> next_event();
(nil)
Gamma>

**See Also**

register_point, when_echo_fns, when_exception_fns, remove_echo_function, remove_exception_function
lock_point

lock_point — locks or unlocks points.

Synopsis

lock_point (symbol, locked)

Arguments

symbol
A point name, as a symbol.

locked
\[ t \] to set a lock, \[ nil \] to release a lock.

Returns

\[ t \] if the function is successful, otherwise \[ nil \].

Description

This function locks or unlocks a point in the Cogent DataHub. The current security level must be greater than or equal to the security level on the point.

Example

Gamma> init_ipc("locker","lq");
\[ t \]
Gamma> write_point(#a,5);
\[ t \]
Gamma> lock_point(#a,t);
\[ t \]
Gamma> write_point(#a,300);
\[ nil \]
Gamma> lock_point(#a,nil);
\[ t \]
Gamma> write_point(#a,300);
\[ t \]
Gamma>

See Also

set_security, point_locked
point_locked

point_locked — indicates if a point is locked.

Synopsis

point_locked (symbol)

Arguments

symbol
A point name, as a symbol.

Returns

t if locked, or nil if not locked.

Example

Gamma> lock_point(#f,t);
t
Gamma> next_event();
nil
Gamma> point_locked(#f);
t
Gamma> lock_point(#f,nil);
t
Gamma> next_event();
nil
Gamma> point_locked(#f);
nil
Gamma>

See Also

lock_point
point_nanoseconds

point_nanoseconds — gives the nanoseconds from point_seconds that a point value changed.

Synopsis

point_nanoseconds (symbol)

Arguments

symbol

A point name, as a symbol.

Returns

A number of nanoseconds.

Description

This function returns the number of nanoseconds after point_seconds that a given point's value changed.

Example

Gamma> clock();
938631678
Gamma> write_point(#1,44);
t
Gamma> next_event();
nil
Gamma> point_seconds(#1);
938631693
Gamma> point_nanoseconds(#1);
735100000
Gamma>

See Also

point_seconds
point_seconds

point_seconds — gives the time the point value changed.

Synopsis

point_seconds (symbol)

Arguments

symbol
A point name, as a symbol.

Returns

A time in seconds.

Description

This function returns the time in seconds when a given point's value changed.

Example

Gamma> clock();
938631678
Gamma> write_point(#1,44);
t
Gamma> next_event();
nil
Gamma> point_seconds(#1);
938631693
Gamma>

See Also

point_nanoseconds
point_security

point_security — gives the security level of a point.

Synopsis

point_security (symbol)

Arguments

symbol
A point name, as a symbol.

Returns

The security level.

Example

Gamma> set_security(5);
0
Gamma> secure_point(#f,3);
t
Gamma> point_security(#f);
3
Gamma>

See Also

secure_point, set_security,
**read_existing_point, read_point**

*read_existing_point, read_point — retrieve points.*

**Synopsis**

```plaintext
read_existing_point (symbol)
read_point (symbol)
```

**Arguments**

*symbol*

A point name, as a symbol.

**Returns**

The value of a point in the Cogent DataHub. If the point is unavailable then `nil` is returned. For `read_existing_point`, if the point does not exist then `nil` is returned. For `read_point`, if the point does not exist then the point is created and a default value is returned.

**Description**

These functions makes a call to the DataHub to retrieve the point whose name is the same as the `symbol`. If the point does not exist in the DataHub, `read_existing_point` returns `nil` and does not create the point. `read_point` will create a point in the DataHub if necessary, whose value and confidence are both zero. If the point name is pre-qualified with a domain name and a colon (`:`), this function will search that domain's data rather than the DataHub for the default domain.

**Example**

This example uses data points entered in the `write_point` reference entry example.

```
Gamma> init_ipc("reader","rq");
Gamma> read_point(#my);
600
Gamma> read_point(#dog);
130
Gamma> read_point(#has);
140
Gamma> read_point(#fleas);
150
Gamma> read_existing_point(#cat);
```

nil
Gamma> read_point(#cat);
0
Gamma>

See Also

register_point, write_point
register_all_points

register_all_points — registers an application to receive exceptions for all points.

Synopsis

```plaintext
register_all_points (domain?, newflag?)
```

Arguments

domain
The Cogent DataHub domain in which to register.

newflag
A flag determining whether to automatically register all future points from the DataHub.

Returns

t on success, or nil on failure.

Description

This function registers the current application to receive exceptions from the Cogent DataHub for all points in the given domain. Once this function has been called, any changes to the value of any point in the DataHub will be transmitted to the input queue of the application. These changes are events, and as such must be processed by calling `next_event` or `next_event_nb` before the application will recognize the new value of the point.

If the domain is nil, then the current default domain (set by `set_domain`) will be used. If the domain is named, even if it is the default domain, then the DataHub will transmit all points as fully qualified names, in the domain:name format. If the `newflag` is given and is non-nil, then any points which are created on the DataHub after this call is made will be automatically registered. If `newflag` is nil or not provided, then the DataHub will not automatically register points which were created since this call.

Example

```plaintext
Gamma> register_all_points(nil,t);
t
Gamma> write_point(#b,22);
t
Gamma> next_event();
nil
```
register_all_points

 Gamma> b;
 22
 Gamma> register_all_points("plant",t);
 t
 Gamma>

See Also

register_point
register_exception

register_exception — not yet documented.

Synopsis

```c
register_exception (symbol, s_exp, execute_p?)
```

Arguments

- `symbol`
- `s_exp`
- `execute_p`

Returns

Description

Example

See Also
register_point, register_existing_point

register_point, register_existing_point — register an application to receive exceptions for a single point.

Synopsis

```
register_point (symbol)
register_existing_point (symbol)
```

Arguments

```symbol```
A point name, as a symbol.

Returns

The current value of the point in the Cogent DataHub. If the point does not exist, `register_point` will create the point in the DataHub and return a default value. However `register_existing_point` will return `nil` if the point does not exist.

Description

These functions register an application to receive changes in the value of a point whenever they occur. The current value of the point is returned as a result of the registration. Once this function has been called, any changes to the value of the point in the DataHub will be transmitted to the input queue of the application. These changes are events, and as such must be processed by calling `next_event` or `next_event_nb` before the application will recognize the new value of the point.

A function may be attached to the value change event using the `when_exception` and `when_echo` functions. Regardless of whether an event is attached to the point, the interpreter will update the value of the symbol whose name is the same as the point name. This means that once a point has been registered its value will always be current in the global scope of the interpreter. If the point name is pre-qualified with a domain name and a colon (:), this function will search that domain’s data rather than the DataHub for the default domain.

Example

```
Gamma> register_point(#f);
26
Gamma> write_point(#f,85);
Gamma> f;
```
See Also

init_ipc, next_event, next_event_nb, when_echo, when_exception
remove_echo_function

remove_echo_function — removes an echo function from a symbol.

Synopsis

remove Echo_function (symbol, echo_fn)

Arguments

symbol
    The point name, as a symbol, from which to remove the echo function.

echo_fn
    The echo function body.

Returns

The echo function which was removed, or nil if no function was removed.

Description

This function removes an echo function (Cogent DataHub echo handler) from the symbol. The echo_fn is compared to all of the current echo functions for the symbol using the comparison function eq.

Example

Gamma> add_echo_function(#temp,#princ("echo\n");
(princ "echo\n")
Gamma> write_point(#temp,28);
(t)
Gamma> next_event();
(t)
Gamma> remove_echo_function(#temp,#princ("echo\n");
(princ "echo\n")
Gamma> write_point(#temp,32);
(t)
Gamma> next_event();
(t)
Gamma>

See Also

add_echo_function
remove_exception_function

remove_exception_function — removes an exception function from a symbol.

Synopsis

remove_exception_function (symbol, exc_fn)

Arguments

symbol
The point name, as a symbol, from which to remove the exception function.

exc_fn
The exception function body.

Returns

The exception function which was removed, or nil if no function was removed.

Description

This function removes an exception function (Cogent DataHub exception handler) from the symbol. The exc_fn is compared to all of the current exception functions for the symbol using the comparison function eq.

Example

Gamma> add_exception_function(#temp, #princ("temp change\n"));
(princ "temp change\n")
Gamma> next_event();
temp change
(t)
Gamma> temp;
t 40
Gamma> remove_exception_function(#temp, #princ("temp change\n"));
(princ "temp change\n")
Gamma> next_event();
nil
Gamma> temp;
t 35
Gamma>

See Also

add_exception_function
secure_point

secure_point — alters the security level on a point.

Synopsis

secure_point (symbol, security)

Arguments

symbol
  The point to alter, as a symbol.

security
  The new security level for this point.

Returns

t  on success, or nil if an error occurred.

Description

This function alters the security level on a point in the Cogent DataHub. If the current process security level is lower than the named point (symbol), then the function returns nil, otherwise it returns t. The initial security level for a process is 0.

Example

Gamma> init_ipc("spt","spq");
t
Gamma> secure_point(#d,5);
nil
Gamma> set_security(9);
0
Gamma> secure_point(#d,5);
t
Gamma> secure_point(#d,12);
nil
Gamma> set_security(15);
9
Gamma> secure_point(#d,12);
t
Gamma>
See Also

point_security, set_security


**set_domain**

set_domain — sets the default domain for future calls.

**Synopsis**

```plaintext
set_domain (domain_name)
```

**Arguments**

*domain_name*

A string.

**Returns**

The *domain_name* argument.

**Description**

This function sets the default Cogent DataHub domain for all future calls to `read_point`, `read_existing_point`, `register_point`, `register_existing_point`, `write_point` and `write_existing_point`. The default domain can be overridden by explicitly placing the domain name at the beginning of the point name, separated by a colon (:). For example, a variable named `tank_level` in the default domain would have to be named `acme:tank_level` in the "acme" domain.

There is a possibility of aliasing points. If the default domain is "acme" then the point `tank_level` and the point `acme:tank_level` refer to the same DataHub point. If both of these names are used in a Gamma program then one of them will not behave correctly. It is the responsibility of the programmer to ensure that there is no aliasing in the assigned names, either by always explicitly naming a point's domain or by programming carefully.

**Example**

```plaintext
Gamma> set_domain("acme");
Gamma> read_point(#tank_level);
12.5
Gamma> set_domain("steamplant");
Gamma> read_point("tank_level");
4.35
Gamma> read_point("acme:tank_level");
12.5
```
See Also

read_point, read_existing_point, register_point, write_point, write_existing_point
set_security

set_security — changes the security level for the current process.

Synopsis

set_security (security_level)

Arguments

security_level

The new security level for this process.

Returns

The previous security level for this process.

Description

This function changes the security level for the current process to the given value. There is no restriction on the security level argument. A low-security process can alter its own security level to be higher.

If it is necessary to have a process's security level to be unalterable, then the set_security function can be re-bound after the security level is originally set (see second example). The only use of security level is in conjunction with the Cogent DataHub.

Example

Gamma> init_ipc("spt","spq");
t
Gamma> secure_point(#d,5);
nil
Gamma> set_security(9);
0
Gamma> secure_point(#d,5);
t
Gamma> secure_point(#d,12);
nil
Gamma> set_security(15);
9
Gamma> secure_point(#d,12);
t
Gamma>
The example below sets the current process's security to 5, and then re-binds `set_security` so that the program can no longer alter its security. The `list` function is used in the re-binding, as it will accept any number of arguments without error, and will have no side-effects.

```
Gamma> set_security(5);
0
Gamma> set_security = list;
(defun list (&optional &rest s_exp...) ...)
Gamma> set_security(9);
(9)
Gamma> secure_point(#g,10);
nil
Gamma> secure_point(#g,6);
nil
Gamma> secure_point(#g,4);
t
Gamma>
```

**See Also**

`point_security`, `secure_point`
unregister_point

unregister_point — stops echo and exception message sending.

Synopsis

\[
\text{unregister_point} \ (\text{symbol})
\]

Arguments

\[
\text{symbol}
\]

The point to unregister.

Returns

\( t \) on success, or \text{nil} on failure.

Description

This function causes the Cogent DataHub to immediately stop sending echo and exception messages for the named point. It is possible that exceptions and echos which are queued to the task will arrive after this function is called, but the DataHub will not generate any new messages.

Example

```
Gamma> \text{register\_all\_points}(\text{nil}, t);
  t
Gamma> b;
  55
Gamma> \text{unregister\_point} (#b);
  t
Gamma> \text{write\_point} (#b, 33);
  t
Gamma> \text{write\_point} (#c, 77);
  t
Gamma> \text{next\_event} ();
  \text{nil}
Gamma> b;
  55
Gamma> c;
  77
Gamma>
```
See Also

register_point, register_all_points
**when_echo_fns, when_exception_fns**

*when_echo_fns, when_exception_fns* — indicate the functions for echos or exceptions on a point.

**Synopsis**

```lisp
when_echo_fns (symbol)
when_exception_fns (symbol)
```

**Arguments**

*symbol*

A point name, as a symbol.

**Returns**

A list of expressions to be evaluated when an echo or exception occurs on the *symbol*.

**Example**

```
Gamma> add_echo_function(#temp,#princ("echo\n"));
(princ "echo\n")
Gamma> add_echo_function(#temp,#temp/2);
(/ temp 2)
Gamma> when_echo_fns(#temp);
((/ temp 2) (princ "echo\n") t)
Gamma> write_point(#temp,22);
t
Gamma> next_event();
echo
(11 t t)
Gamma>
```

**See Also**

*add_echo_function, add_exception_function, remove_echo_function, remove_exception_function*
writeExistingPoint, writePoint

writeExistingPoint, writePoint — write point values.

Synopsis

```plaintext
writeExistingPoint (symbol, value, seconds?, nanoseconds?)
writePoint (symbol, value, seconds?, nanoseconds?)
```

Arguments

- `symbol`  
  A point name, as a symbol.
- `value`  
  Any numeric or string expression.
- `seconds`  
  Number of seconds since Jan 1st, 1970.
- `nanoseconds`  
  Number of nanoseconds within the second.

Returns

t on success or `nil` on failure.

Description

These functions write a point value to the Cogent DataHub. If the point does not exist in the DataHub, `writePoint` will create the point and set its value. `writeExistingPoint` will return `nil` if the point does not exist.

Example

```plaintext
Gamma> init_ipc("writer","wq");
t
Gamma> writeExistingPoint("my",150);
nil
Gamma> writePoint(#my,120);
t
Gamma> writePoint(#dog,130,450000000);
t
Gamma> writePoint("has",140);
t
Gamma> writePoint("fleas",150,1210947,2134444);
```
These points can be viewed in the DataHub (sorted in alphabetical order) using the `dhview` command at the shell prompt:

<table>
<thead>
<tr>
<th>#</th>
<th>Point Name</th>
<th>Conf Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dog</td>
<td>100 130</td>
</tr>
<tr>
<td>2</td>
<td>fleas</td>
<td>100 150</td>
</tr>
<tr>
<td>3</td>
<td>has</td>
<td>100 140</td>
</tr>
<tr>
<td>4</td>
<td>my</td>
<td>100 600</td>
</tr>
</tbody>
</table>

See Also

`read_point, read_existing_point, register_point`