



Documentation Library

Cogent DataHub®

Version 7.3

Cogent Real-Time Systems, Inc.

March 22, 2013

Cogent DataHub®: Version 7.3

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.

Published March 22, 2013
Cogent Real-Time Systems, Inc.
162 Guelph Street, Suite 253
Georgetown, Ontario
Canada, L7G 5X7

Toll Free: 1 (888) 628-2028
Tel: 1 (905) 702-7851
Fax: 1 (905) 702-7850

Information Email: info@cogent.ca
Tech Support Email: support@cogent.ca
Web Site: www.cogent.ca

Copyright © 1995-2013 by Cogent Real-Time Systems, Inc.

Revision History

Revision 7.3-1 September 2010
DataHub WebView, Historian, Redundancy, and OPC A&E.
Revision 6.4 September 2007
Web server, Data Logging and Email/SMS interfaces.
Revision 6.2-1 September 2005
Initial release of documentation.

Copyright, trademark, and software license information.

Copyright Notice

© 1995-2013 Cogent Real-Time Systems, Inc. All rights reserved.

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written consent of Cogent Real-Time Systems, Inc.

Cogent Real-Time Systems, Inc. assumes no responsibility for any errors or omissions, nor do we assume liability for damages resulting from the use of the information contained in this document.

Trademark Notice

Cascade DataHub, Cascade Connect, Cascade DataSim, Connect Server, Cascade Historian, Cascade TextLogger, Cascade NameServer, Cascade QueueServer, RightSeat, SCADALisp and Gamma are trademarks of Cogent Real-Time Systems, Inc.

All other company and product names are trademarks or registered trademarks of their respective holders.

END-USER LICENSE AGREEMENT FOR COGENT SOFTWARE

IMPORTANT - READ CAREFULLY: This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and Cogent Real-Time Systems Inc. ("Cogent") of 162 Guelph Street, Suite 253, Georgetown, Ontario, L7G 5X7, Canada (Tel: 905-702-7851, Fax: 905-702-7850), from whom you acquired the Cogent software product(s) ("SOFTWARE PRODUCT" or "SOFTWARE"), either directly from Cogent or through one of Cogent's authorized resellers.

The SOFTWARE PRODUCT includes computer software, any associated media, any printed materials, and any "online" or electronic documentation. By installing, copying or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. If you do not agree with the terms of this EULA, Cogent is unwilling to license the SOFTWARE PRODUCT to you. In such event, you may not use or copy the SOFTWARE PRODUCT, and you should promptly contact Cogent for instructions on return of the unused product(s) for a refund.

SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

1. **EVALUATION USE:** This software is distributed as "Free for Evaluation", and with a per-use royalty for Commercial Use, where "Free for Evaluation" means to evaluate Cogent's software and to do exploratory development and "proof of concept" prototyping of software applications, and where "Free for Evaluation" specifically excludes without limitation:

- i. use of the SOFTWARE PRODUCT in a business setting or in support of a business activity,
- ii. development of a system to be used for commercial gain, whether to be sold or to be used within a company, partnership, organization or entity that transacts commercial business,
- iii. the use of the SOFTWARE PRODUCT in a commercial business for any reason other than exploratory development and "proof of concept" prototyping, even if the SOFTWARE PRODUCT is not incorporated into an application or product to be sold,
- iv. the use of the SOFTWARE PRODUCT to enable the use of another application that was developed with the SOFTWARE PRODUCT,
- v. inclusion of the SOFTWARE PRODUCT in a collection of software, whether that collection is sold, given away, or made part of a larger collection.
- vi. inclusion of the SOFTWARE PRODUCT in another product, whether or not that other product is sold, given away, or made part of a larger product.

2. **COMMERCIAL USE:** COMMERCIAL USE is any use that is not specifically defined in this license as EVALUATION USE.

3. **GRANT OF LICENSE:** This EULA covers both COMMERCIAL and EVALUATION USE of the SOFTWARE PRODUCT. Either clause (A) or (B) of this section will apply to you, depending on your actual use of the SOFTWARE PRODUCT. If you have not purchased a license of the SOFTWARE PRODUCT from Cogent or one of Cogent's authorized resellers, then you may not use the product for COMMERCIAL USE.

- A. **GRANT OF LICENSE (EVALUATION USE):** This EULA grants you the following non-exclusive rights when used for EVALUATION purposes:

Software: You may use the SOFTWARE PRODUCT on any number of computers, either stand-alone, or on a network, so long as every use of the SOFTWARE PRODUCT is for EVALUATION USE. You may reproduce the SOFTWARE PRODUCT, but only as reasonably required to install and use it in accordance with this LICENSE or to follow your normal back-up practices.

Subject to the license expressly granted above, you obtain no right, title or interest in or to the SOFTWARE PRODUCT or related documentation, including but not limited to any copyright, patent, trade secret or other proprietary rights therein. All whole or partial copies of the SOFTWARE PRODUCT remain property of Cogent and will be considered part of the SOFTWARE PRODUCT for the purpose of this EULA.

Unless expressly permitted under this EULA or otherwise by Cogent, you will not:

- i. use, reproduce, modify, adapt, translate or otherwise transmit the SOFTWARE PRODUCT or related components, in whole or in part;
- ii. rent, lease, license, transfer or otherwise provide access to the SOFTWARE PRODUCT or related components;
- iii. alter, remove or cover proprietary notices in or on the SOFTWARE PRODUCT, related documentation or storage media;
- iv. export the SOFTWARE PRODUCT from the country in which it was provided to you by Cogent or its authorized reseller;
- v. use a multi-processor version of the SOFTWARE PRODUCT in a network larger than that for which you have paid the corresponding multi-processor fees;
- vi. decompile, disassemble or otherwise attempt or assist others to reverse engineer the SOFTWARE PRODUCT;
- vii. circumvent, disable or otherwise render ineffective any demonstration time-outs, locks on functionality or any other restrictions on use in the SOFTWARE PRODUCT;
- viii. circumvent, disable or otherwise render ineffective any license verification mechanisms used by the SOFTWARE PRODUCT;
- ix. use the SOFTWARE PRODUCT in any application that is intended to create or could, in the event of malfunction or failure, cause personal injury or property damage; or
- x. make use of the SOFTWARE PRODUCT for commercial gain, whether directly, indirectly or incidentally.

B. GRANT OF LICENSE (COMMERCIAL USE): This EULA grants you the following non-exclusive rights when used for COMMERCIAL purposes:

Software: You may use the SOFTWARE PRODUCT on one computer, or if the SOFTWARE PRODUCT is a multi-processor version - on one node of a network, either: (i) as a development systems for the purpose of creating value-added software applications in accordance with related Cogent documentation; or (ii) as a single run-time copy for use as an integral part of such an application. This includes reproduction and configuration of the SOFTWARE PRODUCT, but only as reasonably required to install and use it in association with your licensed processor or to follow your normal back-up practices.

Storage/Network Use: You may also store or install a copy of the SOFTWARE PRODUCT on one computer to allow your other computers to use the SOFTWARE PRODUCT over an internal network, and distribute the SOFTWARE PRODUCT to your other computers over an internal network. However, you must acquire and dedicate a license for the SOFTWARE PRODUCT for each computer on which the SOFTWARE PRODUCT is used or to which it is distributed. A license for the SOFTWARE PRODUCT may not be shared or used concurrently on different computers.

Subject to the license expressly granted above, you obtain no right, title or interest in or to the SOFTWARE PRODUCT or related documentation, including but not limited to any copyright, patent, trade secret or other proprietary rights therein. All whole or partial copies of the SOFTWARE PRODUCT remain property of Cogent and will be considered part of the SOFTWARE PRODUCT for the purpose of this EULA.

Unless expressly permitted under this EULA or otherwise by Cogent, you will not:

- i. use, reproduce, modify, adapt, translate or otherwise transmit the SOFTWARE PRODUCT or related components, in whole or in part;

- ii. rent, lease, license, transfer or otherwise provide access to the SOFTWARE PRODUCT or related components;
- iii. alter, remove or cover proprietary notices in or on the SOFTWARE PRODUCT, related documentation or storage media;
- iv. export the SOFTWARE PRODUCT from the country in which it was provided to you by Cogent or its authorized reseller;
- v. use a multi-processor version of the SOFTWARE PRODUCT in a network larger than that for which you have paid the corresponding multi-processor fees;
- vi. decompile, disassemble or otherwise attempt or assist others to reverse engineer the SOFTWARE PRODUCT;
- vii. circumvent, disable or otherwise render ineffective any demonstration time-outs, locks on functionality or any other restrictions on use in the SOFTWARE PRODUCT;
- viii. circumvent, disable or otherwise render ineffective any license verification mechanisms used by the SOFTWARE PRODUCT, or
- ix. use the SOFTWARE PRODUCT in any application that is intended to create or could, in the event of malfunction or failure, cause personal injury or property damage.

4. **WARRANTY:** Cogent cannot warrant that the SOFTWARE PRODUCT will function in accordance with related documentation in every combination of hardware platform, software environment and SOFTWARE PRODUCT configuration. You acknowledge that software bugs are likely to be identified when the SOFTWARE PRODUCT is used in your particular application. You therefore accept the responsibility of satisfying yourself that the SOFTWARE PRODUCT is suitable for your intended use. This includes conducting exhaustive testing of your application prior to its initial release and prior to the release of any related hardware or software modifications or enhancements.

Subject to documentation errors, Cogent warrants to you for a period of ninety (90) days from acceptance of this EULA (as provided above) that the SOFTWARE PRODUCT as delivered by Cogent is capable of performing the functions described in related Cogent user documentation when used on appropriate hardware. Cogent also warrants that any enclosed disk(s) will be free from defects in material and workmanship under normal use for a period of ninety (90) days from acceptance of this EULA. Cogent is not responsible for disk defects that result from accident or abuse. Your sole remedy for any breach of warranty will be either: i) terminate this EULA and receive a refund of any amount paid to Cogent for the SOFTWARE PRODUCT, or ii) to receive a replacement disk.

5. **LIMITATIONS:** Except as expressly warranted above, the SOFTWARE PRODUCT, any related documentation and disks are provided "as is" without other warranties or conditions of any kind, including but not limited to implied warranties of merchantability, fitness for a particular purpose and non-infringement. You assume the entire risk as to the results and performance of the SOFTWARE PRODUCT. Nothing stated in this EULA will imply that the operation of the SOFTWARE PRODUCT will be uninterrupted or error free or that any errors will be corrected. Other written or oral statements by Cogent, its representatives or others do not constitute warranties or conditions of Cogent.

In no event will Cogent (or its officers, employees, suppliers, distributors, or licensors: collectively "Its Representatives") be liable to you for any indirect, incidental, special or consequential damages whatsoever, including but not limited to loss of revenue, lost or damaged data or other commercial or economic loss, arising out of any breach of this EULA, any use or inability to use the SOFTWARE PRODUCT or any claim made by a third party, even if Cogent (or Its Representatives) have been advised of the possibility of such damage or claim. In no event will the aggregate liability of Cogent (or that of Its Representatives) for any damages or claim, whether in contract, tort or otherwise, exceed the amount paid by you for the SOFTWARE PRODUCT.

These limitations shall apply whether or not the alleged breach or default is a breach of a fundamental condition or term, or a fundamental breach. Some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, or certain limitations of implied warranties. Therefore the above limitation may not apply to you.

6. **DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS:**

Separation of Components. The SOFTWARE PRODUCT is licensed as a single product. Its component parts may not be separated for use on more than one computer.

Termination. Without prejudice to any other rights, Cogent may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such an event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts.

7. **UPGRADES:** If the SOFTWARE PRODUCT is an upgrade from another product, whether from Cogent or another supplier, you may use or transfer the SOFTWARE PRODUCT only in conjunction with that upgrade product, unless you destroy the upgraded product. If the SOFTWARE PRODUCT is an upgrade of a Cogent product, you now may use that upgraded product only in accordance with this EULA. If the SOFTWARE PRODUCT is an upgrade of a component of a package of software programs which you licensed as a single product, the SOFTWARE PRODUCT may be used and transferred only as part of that single product package and may not be separated for use on more than one computer.
8. **COPYRIGHT:** All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text and 'applets', incorporated into the SOFTWARE PRODUCT), any accompanying printed material, and any copies of the SOFTWARE PRODUCT, are owned by Cogent or its suppliers. You may not copy the printed materials accompanying the SOFTWARE PRODUCT. All rights not specifically granted under this EULA are reserved by Cogent.
9. **PRODUCT SUPPORT:** Cogent has no obligation under this EULA to provide maintenance, support or training.
10. **RESTRICTED RIGHTS:** Use, duplication, or disclosure by the U.S. Government is subject to restrictions as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (OCT 1988), FAR 12.212(a)(1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as appropriate. Manufacturer is Cogent Real-Time Systems Inc. 162 Guelph Street, Suite 253, Georgetown, Ontario, L7G 5X7, Canada.
11. **GOVERNING LAW:** This Software License Agreement is governed by the laws of the Province of Ontario, Canada. You irrevocably attorn to the jurisdiction of the courts of the Province of Ontario and agree to commence any litigation that may arise hereunder in the courts located in the Judicial District of Peel, Province of Ontario.

Table of Contents

The Cogent DataHub Product Family	??
1. DataHub WebView	i
2. DataHub OPC Tunneller	ii
3. QuickTrend.....	ii
4. DataHub Logger.....	iii
5. DataHub Bridge	iv
6. DataHub System Monitor	iv
7. OPC DataHub	v
8. Cascade DataHub	vi
9. The Cogent DataHub	vii
1. Installation.....	1
1.1. System Requirements and Installation	1
1.2. Installing Licenses.....	1
1.3. Installing the DataHub as a Service	2
1.4. Performing a Silent (Unattended) Install	7
1.5. Installing Version 7.x alongside Version 6.x	7
1.6. Configuration Files.....	10
2. Getting Started.....	13
2.1. Running the Cogent DataHub	13
2.2. Test with simulated data.....	16
2.3. Connect to an OPC server	17
2.4. Connect from an OPC client	23
2.5. Test the Web Server.....	24
2.6. View Your Data on the Web	25
2.7. Test with Excel	25
2.8. Connect to remote data	27
3. OPC Tunnelling.....	29
3.1. Introduction	29
3.2. Configuring the DataHub for the server	30
3.3. Configuring the Cogent DataHub for the client	37
3.4. Testing the connection.....	40
3.5. Tunnelling part of the data set.....	41
3.6. Extended applications	41
3.6.1. Tunnelling and Bridging.....	??
3.6.2. Tunnelling and Aggregation	??
4. OPC Aggregation.....	44
4.1. Introduction	44
4.2. Configuring the DataHub	44
4.3. Extended applications	51
4.3.1. Aggregation and Bridging	51
4.3.2. Aggregation and Tunnelling	51
5. OPC Bridging.....	53
5.1. Introduction	53
5.2. Configuring Bridges	53
5.2.1. Point-to-point configuration	53
5.2.2. Making transformations.....	55
5.3. Creating New Points	57
5.4. Bridging Scenarios	58

5.4.1. Bridging Local Servers	58
5.4.2. Bridging Remote Servers	58
5.4.3. Creating Data Sets	59
5.4.4. Bridging to Excel.....	59
6. Excel Connections.....	61
6.1. Getting Data into Excel.....	61
6.1.1. Method 1 - Drag and Drop using DDEAdvise	61
6.1.2. Method 2 - Excel Macros using DDERequest	63
6.1.2.1. Create a macro	??
6.1.2.2. Add a Control Button.....	63
6.1.2.3. Receive the data	??
6.2. Getting Data out of Excel.....	64
6.2.1. Method 1 - Configuring DDEAdvise loops in the Cogent DataHub	65
6.2.2. Method 2 - Writing Excel macros that use the DDEPoke command	67
6.2.2.1. Create a macro	??
6.2.2.2. Add a Control Button.....	68
6.2.2.3. Send the data.....	??
6.2.2.4. Additional Pointers	68
6.3. Example	69
6.4. Networking Excel.....	71
6.5. Working with Ranges	74
6.5.1. Getting a Range out of Excel.....	75
6.5.2. Getting a Range into Excel.....	76
6.5.3. Sample Excel Macros for Arrays	78
7. Using QuickTrend.....	80
8. Using the Web Server	84
8.1. Introduction	84
8.2. Configuring the Web Server.....	86
8.3. Viewing the Web Demos.....	88
8.4. Viewing Your Own Data	89
8.4.1. DataHub Browser	89
8.4.2. DataHub Table View.....	90
8.5. Modifying the ASP and AJAX Demo Files	91
8.6. Using ASP to Query a Database and Display Results	94
8.7. Generating and Receiving XML with the Cogent DataHub	96
8.7.1. Streaming XML How-To.....	97
8.7.1.1. Built-in Streaming Data	??
8.7.1.2. Customizing the Built-in Streaming Data	??
8.7.2. Polling XML How-To.....	102
8.7.2.1. Built-in HTTP Data Polling	??
8.7.2.2. Customized HTTP Data Polling	??
9. Write to a Database	105
9.1. Introduction	105
9.2. Quick Start	105
9.3. Configuring the Queue, Store and Forward	109
9.4. Setting up the DSN (Data Source Name).....	111
9.5. Configuring a Database Table	113
9.6. Key Columns.....	117
9.7. Assigning a Trigger.....	118
9.8. Setting Trigger Conditions	119

9.9. Configured Actions	124
10. Query a Database.....	126
10.1. Introduction.....	126
10.2. Quick Start	126
10.3. Setting up the DSN (Data Source Name).....	129
10.4. Configuring a Database Query	131
10.5. Assigning a Trigger.....	134
10.6. Setting Trigger Conditions	136
10.7. Configured Actions	139
11. System Monitor	141
11.1. Introduction	141
11.2. Configuring the System Monitor.....	141
11.3. Monitoring Systems Across a Network	144
12. Email and SMS	148
12.1. Introduction	148
12.1.1. How it works	148
12.1.2. A note about SMS text messages	148
12.2. Configuring the Mail Server.....	149
12.3. Sending a Test Message	150
12.4. Defining the Email Message	154
12.5. Assigning a Trigger.....	156
12.6. Setting Trigger Conditions	158
12.7. Configured Actions	161
12.8. HTML Message Examples.....	161
12.8.1. An HTML Message with Embedded Data Points	161
12.8.2. An HTML Message with a Table Created in Code	163
12.9. Dynamically Changing Email Subjects and Recipients.....	165
13. Linux Connections	166
13.1. Configuring the Cogent DataHub in Windows	166
13.2. Configuring the Cascade DataHub on Linux	170
14. QNX Connections	172
14.1. Configuring the Cogent DataHub in Windows	172
14.2. Configuring the Cascade DataHub on QNX	176
15. InTouch Connections	178
15.1. Getting data into InTouch.....	178
15.2. Getting data out of InTouch	180
15.3. Reading and writing data in both directions	181
15.4. Error message displayed when starting InTouch WindowViewer.....	181
16. DataHub Scripting.....	183
16.1. Tools	183
16.2. DataHub ODBC (Open Database Connectivity) Scripting	183
16.3. DataHub Windows Scripting.....	184
17. Security	186
17.1. How to Configure	??
17.2. SSL and Firewalls	188
17.3. User Authentication	188
17.4. Authorization and User Permissions.....	190
17.5. Permissions for the DataHub Command Set.....	193

17.6. Passwords	196
18. Working With Data	197
18.1. Data Points	197
18.1.1. Creating New Points	??
18.1.2. Deleting Points	??
18.1.3. Viewing Data Points	197
18.1.4. Point Size Limits	197
18.2. Data Communication Concepts	198
18.2.1. Send and Receive Data	198
18.2.2. Client - Server	198
18.2.3. Synchronous and Asynchronous Communication	198
18.3. Data Exchange Protocols	199
18.3.1. OPC Protocol	199
18.3.2. DDE Protocol	200
18.3.3. TCP and Tunnelling/Mirroring	200
18.3.4. The DataHub API	200
18.4. Data Organization	201
18.4.1. Data Domains	201
18.4.2. Assemblies, Subassemblies, Attributes, and Properties	201
18.4.3. Attributes and Types	202
18.4.4. Example 1: Attributes and Types	202
18.4.5. Example 2: Private Attributes	204
19. Optimizing Data Throughput	206
19.1. Binary Mode Tunnel/Mirror (TCP) Connections	206
19.2. Tunnel/Mirror (TCP) Connections for Slow Networks	206
19.3. Old Value Queuing	206
19.4. Un-Buffered Delivery	207
19.5. Screen Output	207
19.6. CPU Saturation	208
19.7. How to Optimize	208
19.7.1. Tunnel/Mirror (TCP) connections	208
19.7.2. DataHub C++ API	210
19.7.3. Gamma scripts	210
20. Properties Window	212
20.1. General	212
20.2. OPC DA	213
20.3. OPC A&E	220
20.4. Tunnel/Mirror	223
20.5. Bridging	227
20.6. DDE	231
20.7. Quick Trend	233
20.8. DataHub WebView	237
20.9. Web Server	238
20.10. Database	240
20.11. Historian	244
20.12. System Monitor	247
20.13. Email/SMS	250
20.14. Redundancy	251
20.14.1. Warm Standby	256
20.15. Scripting	257

20.16. Security	259
20.17. Licenses.....	263
21. Other Windows and Programs	266
21.1. Data Browser.....	266
21.2. Connection Viewer.....	267
21.3. Event Log	269
21.4. Script Editor	270
21.5. Script Log.....	270
21.6. DataSim - a data simulation program	271
21.7. DataPid - a PID loop data simulation program	273
21.8. Service Manager	276
22. Using DataHub Commands	282
22.1. Command Syntax	282
22.2. Return Syntax.....	282
22.3. Sending Commands by TCP	283
23. Troubleshooting	285
A. Command Line Options	288
B. Excel Macro Library	290
B.1. Configure Excel to receive data from the Cogent DataHub (using DDEAdvise)	290
B.2. Write data from Excel - User initiated (using DDEPoke).....	291
B.3. Write data from Excel - Automatically on value change (using DDEPoke).....	292
B.4. Other Useful Macros	293
C. Running as a Windows Service (Specified User).....	294
D. Windows Services File for Tunnel/Mirror	296
E. OPC Overview	297
F. DDE Overview	299
G. ODBC Database Concepts	302
H. Error Messages	304
H.1. Windows Error Numbers.....	304
H.2. Windows TCP Error Numbers	305
H.3. Windows DDE Error Numbers	306
I. Third-Party Source Licenses.....	308
J. Media Source Licenses	310
K. GNU General Public License.....	311
L. GNU Lesser General Public License	317
I. Cogent DataHub Command Set	325
acksuccess	328
add	329
alias	330
alive.....	331
append	332
assembly	333
asyncsocket.....	334
attribute.....	335
auth	336
authgroup.....	337
authuser.....	338
auto_create_domains	339

auto_timestamp	340
bridge.....	341
bridge_remove	343
bridge_remove_pattern	344
bridge_transform	345
cforce	347
cread	348
create	349
create_domain	350
creport	351
cset	352
cwrite	353
DDEAdvise.....	355
DDEConnect.....	356
DDEDisconnect.....	357
DDEInit	358
DDEService	359
DDEUnadvise	360
DDEUnadvisePattern.....	361
DDEUnadvisePoint	362
debug	363
defaultprop.....	364
delete.....	365
deleted	366
div	367
domain.....	368
domains	369
dump.....	370
enable_bridging.....	371
enable_dde_client	372
enable_dde_server.....	373
enable_mirror_master	374
enable_mirror_slave.....	375
enable_opc_client	376
enable_opc_server	377
enable_scripting	378
enable_tcp_server.....	379
error.....	380
execute_plugin	381
exit	382
flush	383
flush_log	384
force	385
format.....	386
heartbeat	387
ignore.....	388
ignore_old_data	389
include	390
instance.....	391
load_config_files	392
load_plugin	393

load_scripts	394
lock	395
log_file	396
log_to_file	397
mirror_master	398
mirror_master_2	399
mult	401
OPCActivate	402
OPCAddItem2	403
OPCApply	404
OPCAttach2	405
OPCConnect	407
OPCDetach	408
OPCEnable	409
OPCEnableClient	410
OPCEnableServer	411
OPCMinimumSecurity	412
OPCModify	413
OPCRefresh	415
OPCReload	416
OPCRemoveItem	417
private_attribute	418
property	419
quality	420
read	421
report	422
report_domain	423
report_errors	424
request_initial_data	425
save_config	426
secure	427
set	428
set_canonical	429
show_data	430
show_debug_messages	431
show_event_log	432
show_icon	433
show_properties	434
show_script_log	435
subassembly	436
success	437
tcp_service	438
timeout	439
transmit_insignificant	440
type	441
unload_plugin	442
unreport	443
version	444
write	445
II. Obsolete and Unused Commands	447
bandwidth_reduce	448

drop_license	449
echo	450
enable_connect_server	451
EnableDDEServer	452
exception_buffer	453
failed_license	454
master_host	455
master_service	456
on_change	457
OPCAddItem	458
OPCAttach	459
OPCInit	460
point	461
qnx_name_attach	462
qnx_receiver	463
readid	464
register_datahub	465
report_all	466
report_datahubs	467
request	468
run	469
script_register	470
script_symbol	471
slave	472
sync	473
taskdied	474
taskstarted	475
using_license	476
warn_of_license_expiry	477
Index	??
Colophon	484

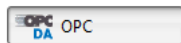
The Cogent DataHub Product Family

The Cogent DataHub is available as a family of related products that provide commonly requested DataHub options. Each of these products can be customized, if desired, by adding any other option shown in the [DataHub Properties window](#).

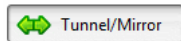
1. DataHub WebView

DataHub WebView is a web-based data visualization tool that features the Cogent DataHub as a back-end data delivery platform and a browser-based editor for designing animated displays of DataHub data that can be viewed using a standard web browser from anywhere on the Internet or corporate network. It can network with other network-enabled Cogent DataHub products, and exchange data in real time.

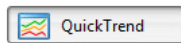
For the DataHub WebView Manual: Click here.



The [OPC option](#) lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).



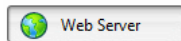
The [Tunnel/Mirror option](#) lets you configure the Cogent DataHub to act as a master or slave for *tunnelling/mirroring*. Tunnelling/Mirroring allows you to send OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.



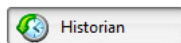
The [QuickTrend option](#) lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.



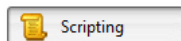
The [DataHub WebView option](#) is a web-based data visualization tool that provides a browser-based editor for designing animated displays of DataHub data that can be viewed using a standard web browser from anywhere on the Internet or corporate network.



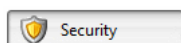
The [Web Server option](#) lets you configure the the DataHub to run as a lightweight http server capable of serving HTML documents, Java applets, and many kinds of binary files. It features password-protected access and server-side scripting, and supports [DataHub WebView](#).



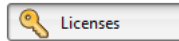
The [Historian option](#) allows you to collect and store histories for groups of data points. It gets configured automatically by the [Quick Trend](#) option, and can be configured manually as well.



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.



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 [Chapter 17, Security](#).



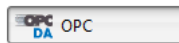
The [Licenses option](#) lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, the Cogent DataHub switches to license mode, and each connection then requires a license.

In addition to these features, WebView can [aggregate data sources](#), and it supports connections from the DataHub APIs for C++, Java, and .NET.

2. DataHub OPC Tunneller

DataHub OPC Tunneller provides robust and secure networking (tunnelling) for OPC DA data, eliminating the headaches of DCOM. It can network with other network-enabled DataHub products, and exchange data in real time.

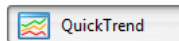
For OPC Tunnelling documentation: [Click here](#).



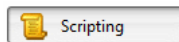
The [OPC option](#) lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).



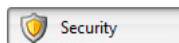
The [Tunnel/Mirror option](#) lets you configure the Cogent DataHub to act as a master or slave for *tunnelling/mirroring*. Tunnelling/Mirroring allows you to send OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.



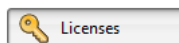
The [QuickTrend option](#) lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.



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.



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 [Chapter 17, Security](#).



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, the Cogent DataHub switches to license mode, and each connection then requires a license.

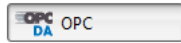
In addition to these features, WebView can [aggregate data sources](#), and it supports connections from the DataHub APIs for C++, Java, and .NET.

3. QuickTrend

QuickTrend is a trending tool based on the Cogent DataHub that lets you create a live trending graph for any number of data points. 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. It

can connect to OPC and DDE servers, and can act as a tunnelling slave to any network-enabled DataHub product.

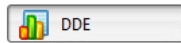
For QuickTrend documentation: [Click here.](#)



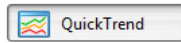
The [OPC option](#) lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).



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 OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.



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 [Section 18.3.2, DDE Protocol](#) and [Appendix F, DDE Overview](#).



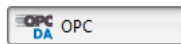
The [QuickTrend option](#) lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.

In addition to these features, WebView can [aggregate data sources](#).

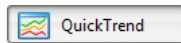
4. DataHub Logger

DataHub Logger provides a quick way to store process data from OPC servers in any ODBC-compliant database, such as Access, SQL Server, Oracle or MySQL.

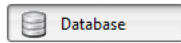
For Data Logging documentation: [Click here.](#)



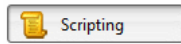
The [OPC option](#) lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).



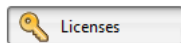
The [QuickTrend option](#) lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.



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



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.



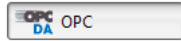
The [Licenses option](#) lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, the Cogent DataHub switches to license mode, and each connection then requires a license.

In addition to these features, WebView can [aggregate data sources](#).

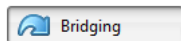
5. DataHub Bridge

DataHub Bridge lets you connect between servers with a few clicks of the mouse. You can link OPC or DDE servers with each other, or bridge other data sources into a single cohesive group. Bridging also provides an easy way to scale or modify the value of a data point as it passes through the DataHub.

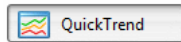
For Bridging documentation: [Click here](#).



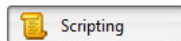
The [OPC option](#) lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).



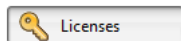
The [Bridging option](#) lets you configure the Cogent DataHub to configure data bridging. Bridging means connecting points from two different DataHub clients so that when one point changes, its value gets written to the bridged point.



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.



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.



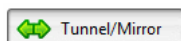
The [Licenses option](#) lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, the Cogent DataHub switches to license mode, and each connection then requires a license.

In addition to these features, WebView can [aggregate data sources](#).

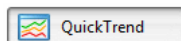
6. DataHub System Monitor

DataHub System Monitor allows you to monitor the performance of a computer anywhere on your network, accessing system parameters such as CPU usage, free disk space, available memory and whether critical processes are still running. It can network with other network-enabled DataHub products, and exchange data in real time.

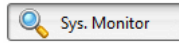
For System Monitor documentation: [Click here](#).



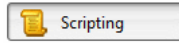
The [Tunnel/Mirror option](#) lets you configure the Cogent DataHub to act as a master or slave for *tunnelling/mirroring*. Tunnelling/Mirroring allows you to send OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.



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.



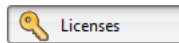
The [System Monitor option](#) allows you to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. with the Cogent DataHub.



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.



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 [Chapter 17, Security](#).

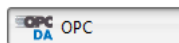


The [Licenses option](#) lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, the Cogent DataHub switches to license mode, and each connection then requires a license.

In addition to these features, WebView can [aggregate data sources](#), and it supports connections from the DataHub APIs for C++, Java, and .NET.

7. OPC DataHub

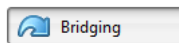
The OPC DataHub is a complete real-time data integration tool for OPC DA applications, including OPC tunnelling and bridging, data logging, and DDE connectivity, as well as scripting, email, and system monitoring. It can network with other network-enabled DataHub products, and exchange data in real time.



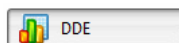
The [OPC option](#) lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).



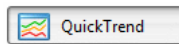
The [Tunnel/Mirror option](#) lets you configure the Cogent DataHub to act as a master or slave for *tunnelling/mirroring*. Tunnelling/Mirroring allows you to send OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.



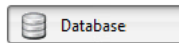
The [Bridging option](#) lets you configure the Cogent DataHub to configure data bridging. Bridging means connecting points from two different DataHub clients so that when one point changes, its value gets written to the bridged point.



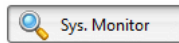
The [DDE option](#) lets you configure the Cogent DataHub to act as a DDE client and/or DDE server for **DDEAdvise** messages. For more information on DDE, please refer to [Section 18.3.2, DDE Protocol](#) and [Appendix F, DDE Overview](#).



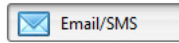
The [QuickTrend option](#) lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.



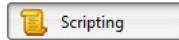
The [Database option](#) lets you configure the DataHub for writing data or making queries to any ODBC-compliant database.



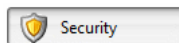
The [System Monitor option](#) allows you to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. with the Cogent DataHub.



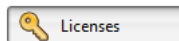
The [Email/SMS option](#) lets you configure the Cogent DataHub to send emails and SMS text messages. The outgoing mail server is configured once, and each email message is configured separately.



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.



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 [Chapter 17, Security](#).

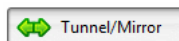


The [Licenses option](#) lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, the Cogent DataHub switches to license mode, and each connection then requires a license.

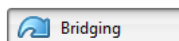
In addition to these features, WebView can [aggregate data sources](#), and it supports connections from the DataHub APIs for C++, Java, and .NET.

8. Cascade DataHub

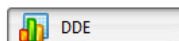
The Cascade DataHub is a complete real-time data integration tool for networking Excel spreadsheets and other DDE-based program. It also provides data logging, scripting, email, and system monitoring. It can network with other network-enabled DataHub products, and exchange data in real time.



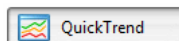
The [Tunnel/Mirror option](#) lets you configure the Cogent DataHub to act as a master or slave for *tunnelling/mirroring*. Tunnelling/Mirroring allows you to send OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.



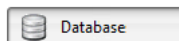
The [Bridging option](#) lets you configure the Cogent DataHub to configure data bridging. Bridging means connecting points from two different DataHub clients so that when one point changes, its value gets written to the bridged point.



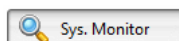
The [DDE option](#) lets you configure the Cogent DataHub to act as a DDE client and/or DDE server for **DDEAdvise** messages. For more information on DDE, please refer to [Section 18.3.2, DDE Protocol](#) and [Appendix F, DDE Overview](#).



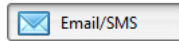
The [QuickTrend option](#) lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.



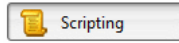
The [Database option](#) lets you configure the DataHub for writing data or making queries to any ODBC-compliant database.



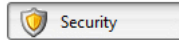
The [System Monitor option](#) allows you to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. with the Cogent DataHub.



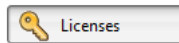
The [Email/SMS option](#) lets you configure the Cogent DataHub to send emails and SMS text messages. The outgoing mail server is configured once, and each email message is configured separately.



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.



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 [Chapter 17, Security](#).

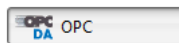


The [Licenses option](#) lets you view and install licenses for the Cogent DataHub. When the Cogent DataHub starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, the Cogent DataHub switches to license mode, and each connection then requires a license.

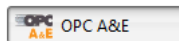
In addition to these features, WebView can [aggregate data sources](#), and it supports connections from the DataHub APIs for C++, Java, and .NET.

9. The Cogent DataHub

The Cogent DataHub is the complete collection of all DataHub features. It incorporates all functionality of all Cogent DataHub family products, and can network with other network-enabled DataHub products, and exchange data in real time.



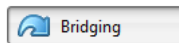
The [OPC option](#) lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).



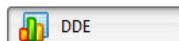
The [OPC A&E option](#) lets you configure the Cogent DataHub to act as an OPC A&E server, an OPC A&E client, or both simultaneously.



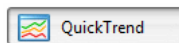
The [Tunnel/Mirror option](#) lets you configure the Cogent DataHub to act as a master or slave for *tunnelling/mirroring*. Tunnelling/Mirroring allows you to send OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.



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.



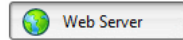
The [DDE option](#) lets you configure the Cogent DataHub to act as a DDE client and/or DDE server for **DDEAdvise** messages. For more information on DDE, please refer to [Section 18.3.2, DDE Protocol](#) and [Appendix F, DDE Overview](#).



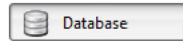
The [QuickTrend option](#) lets you create a live trending graph for any number of data points in any domain of the Cogent DataHub. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.



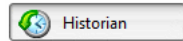
The [DataHub WebView option](#) is a web-based data visualization tool that provides a browser-based editor for designing animated displays of DataHub data that can be viewed using a standard web browser from anywhere on the Internet or corporate network.



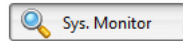
The [Web Server option](#) lets you configure the the DataHub to run as a lightweight http server capable of serving HTML documents, Java applets, and many kinds of binary files. It features password-protected access and server-side scripting, and supports [DataHub WebView](#).



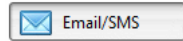
The [Database option](#) lets you configure the DataHub for writing data or making queries to any ODBC-compliant database.



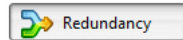
The [Historian option](#) allows you to collect and store histories for groups of data points. It gets configured automatically by the [Quick Trend](#) option, and can be configured manually as well.



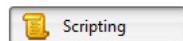
The [System Monitor option](#) allows you to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. with the Cogent DataHub.



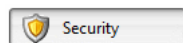
The [Email/SMS option](#) lets you configure the Cogent DataHub to send emails and SMS text messages. The outgoing mail server is configured once, and each email message is configured separately.



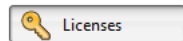
The [Redundancy option](#) lets you configure redundant connections to the Cogent DataHub.



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.



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 [Chapter 17, Security](#).



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, the Cogent DataHub switches to license mode, and each connection then requires a license.

In addition to these features, WebView can [aggregate data sources](#), and it supports connections from the DataHub APIs for C++, Java, and .NET.

Chapter 1. Installation

1.1. System Requirements and Installation

System Requirements

The Cogent DataHub is compatible with Windows 7, 2008, Vista, XP, 2003, for both 32-bit and 64-bit versions.

- Windows 7 (32-bit & 64-bit)
- Windows Server 2008 (32-bit & 64-bit)
- Windows Server 2008 R2 (32-bit & 64-bit)
- Windows Vista (32-bit & 64-bit)
- Windows XP SP2 (32-bit & 64-bit)
- Windows Server 2003 SP2 (32-bit & 64-bit)

Installation

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

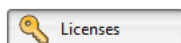
1. Double-click on the program archive `DataHub-7.3-xxxxxx-Windows.exe`.
2. Follow the instructions.

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.

1.2. Installing Licenses

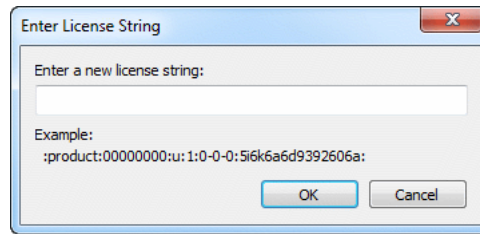
You can install DataHub licenses purchased from Cogent using the **Licenses** option in the Properties window.



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, the Cogent DataHub switches to license mode, and each connection 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 `l` and `1` 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 [Section 1.6, 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.

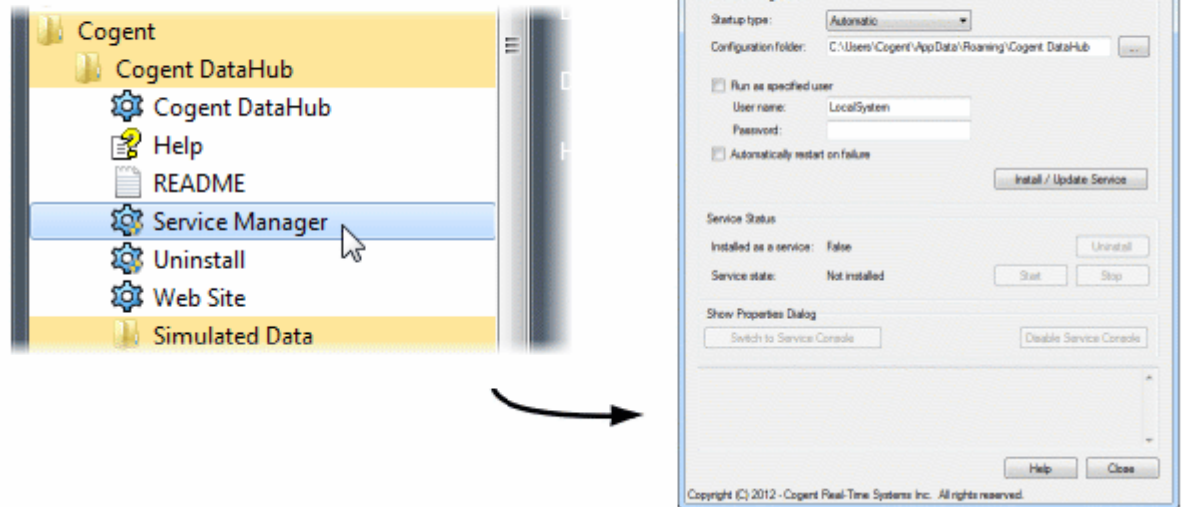
1.3. Installing the DataHub as a Service

The Cogent DataHub can be installed as a service, using the Service Manager. With this program you can select and configure how the DataHub runs, change its status, and open the [Properties window](#) of the Cogent DataHub.

Starting the Service Manager

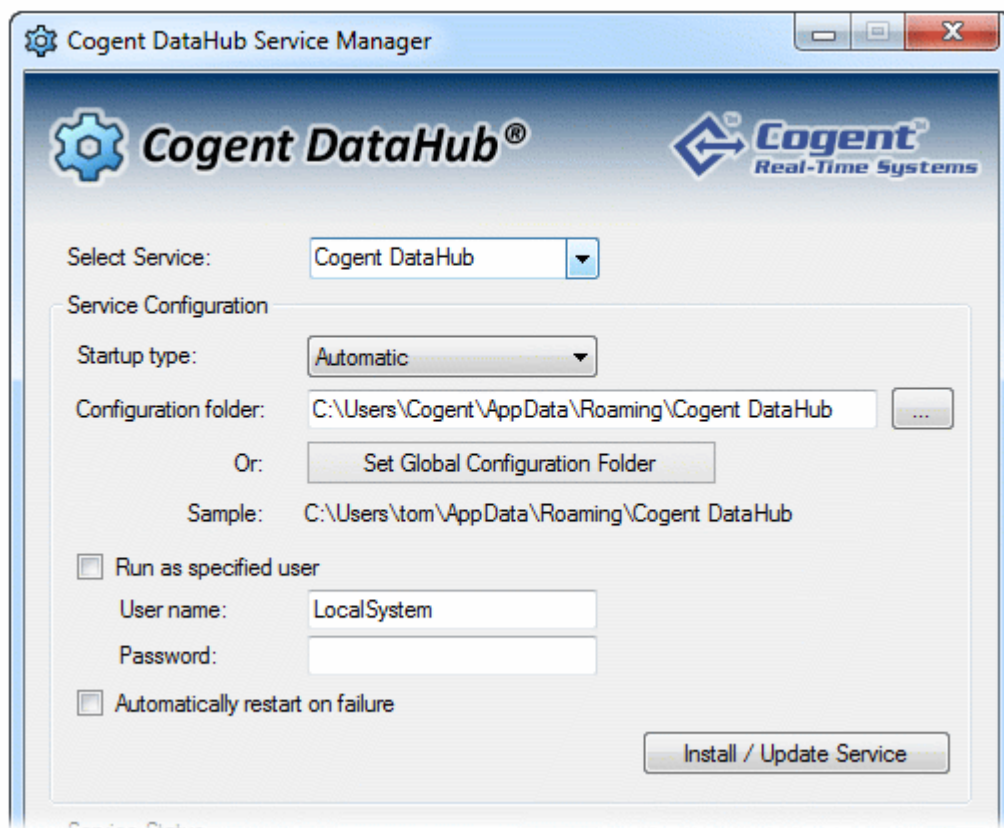
The Service Manager can be started from the Cogent DataHub program group in the Cogent entry of the Windows Start menu.

Select



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 [Section 1.6, Configuration Files](#) for more information about configuration files.

The **Set Global Configuration Folder** button lets you specify the configuration file through the registry. This will preserve the configuration directory even after uninstalling as a service.

Run as specified user

It is recommended that you run the DataHub as the local SYSTEM (LocalSystem) user, and leave this box unchecked.



The DataHub Properties window is only available when running the service as the local SYSTEM user. If you run as a specified user then you will not be able to access the Properties window to make changes to the DataHub while it is running as a service.



However, there are certain situations in which you may need to run it as a specified user. To specify a user other than the local SYSTEM user, enter the **User name** and **Password** as applicable, then please see [Appendix C, Running as a Windows Service \(Specified User\)](#) for important additional information

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**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:

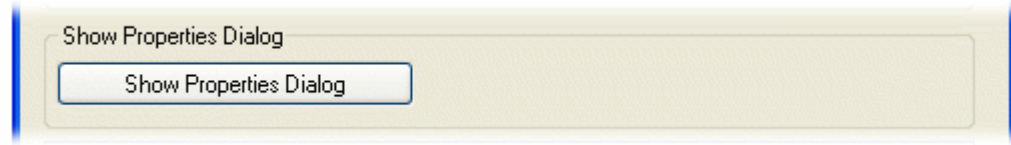
Indicates the run status of the service (Stopped, StartPending, Running, etc.) . The Start and Stop buttons allow you to start or stop the service.

Show Properties Dialog

This option appears differently for different versions of the Windows operating system:

Windows XP and 2003

A Show Properties Dialog button allows you to open the Cogent DataHub Properties window.

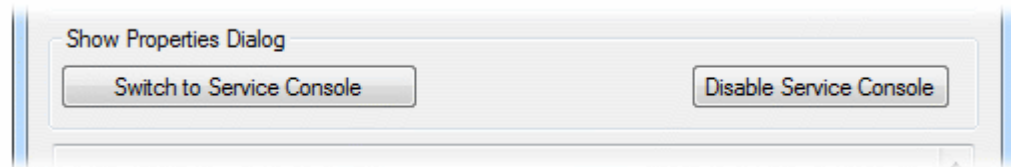


The DataHub Properties window is only visible on the primary console of the computer running the Cogent DataHub. If you are currently logged in via a remote desktop session, you will see a pop-up dialog indicating that you must be connected to the computer's primary console. You can do this by using the `/admin` or `/console` options on the Microsoft Remote Desktop client.

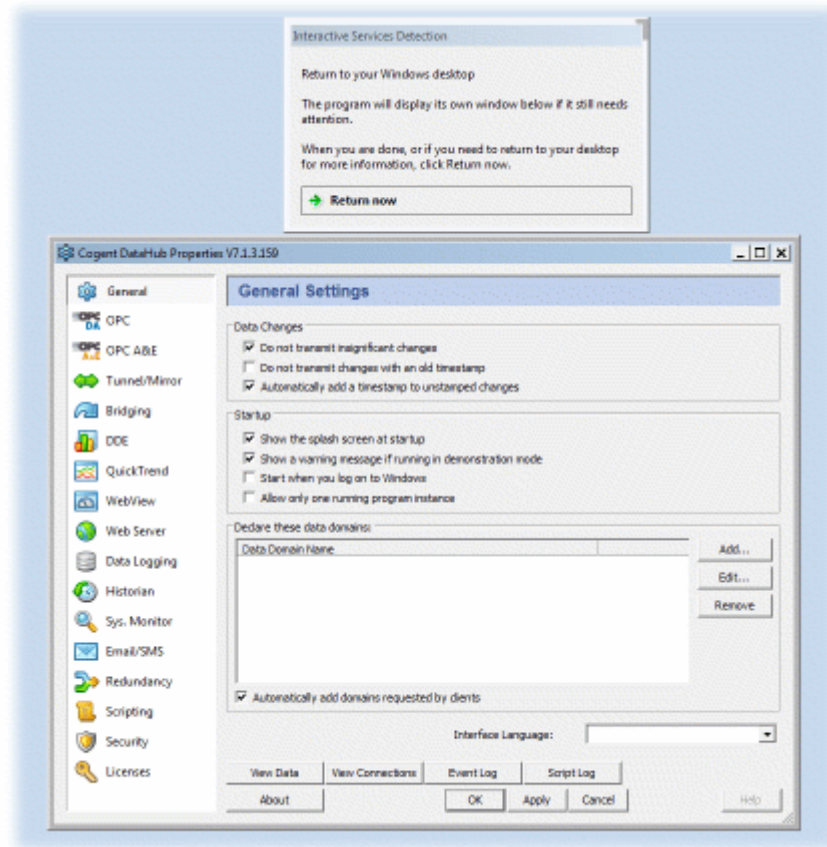
If you are already connected to the primary console, the DataHub Properties window and system tray icon will be displayed when you press Show Properties Dialog button.

Windows Vista, 7 and 2008

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.



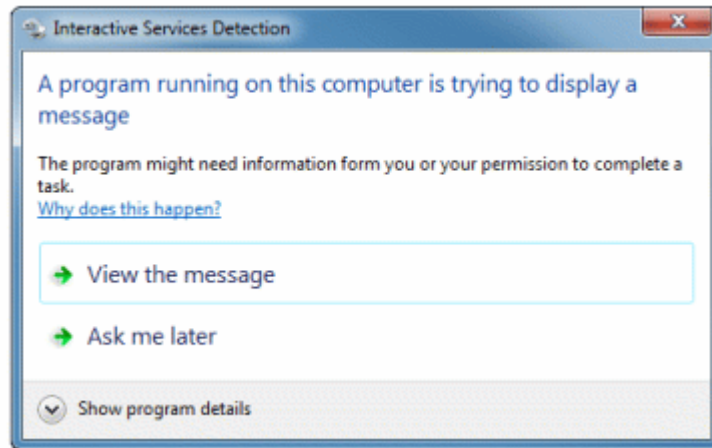
While the Service Console is open, your other applications will continue to run normally, but you will not be able to see or interact with them. A dialog box will be visible while you are viewing the Service Console that will allow you to switch back to your regular desktop at any time. If you have other 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 the DataHub properties, before returning to the normal user console, you should click the Apply and OK buttons to close down the 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.

1.4. 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 /D=c:\program files\cogent
```

/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:

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

1.5. Installing Version 7.x alongside Version 6.x

Installing Cogent DataHub v7.0 and OPC DataHub v6.4 on the same computer

Cogent DataHub v7.0 is designed to install and run side-by-side with OPC DataHub or Cascade DataHub v6.4 on the same computer. To do this, v7.0 maintains its own independent configuration. If you want to carry your v6.4 configuration forward to v7.0, you must copy your existing configuration to the v7.0 configuration directory. When v7.0 first runs, it will ask you whether you want to make a copy of your existing configuration. If you answer **Yes** then your existing configuration will be copied and upgraded to the v7.0 format.



If you copy the configuration, you will need to change some of the defaults. Since OPC DataHub or Cogent DataHub v6.4 and Cogent DataHub v7.0 provide many features in common, their default configurations will conflict when they are run at the same time. In addition, to support the added features of v7.0, the default TCP port values have changed. This can potentially cause the DataHub WebView subsystem in v7.0 to fail until the configuration has been modified.

Even if you choose not to copy the v6.4 configuration, there are some possible conflicts you should know about when running both versions on the same computer.

Affected subsystems in DataHub v7.0

- **Tunnel/Mirror:** Due to limitations placed on TCP port numbers by Microsoft Silverlight, the default port number for the Tunnel/Mirror subsystem has changed from 4600 to 4502.
- **Web Server:** Cogent DataHub version 7.0 installs the updated HTML files and all of the supporting files for DataHub WebView in a directory that is separate from v6.4.
- **Web Server:** Cogent DataHub v7.0 and DataHub v6.4 both attempt to listen to port 80 for HTTP requests.
- **OPC Server:** OPC DataHub and Cogent DataHub register different PROGIDs. OPC client configurations will need to be updated.

Possible conflicts between DataHub v7.0 and v6.4

If you choose to copy the v6.4 configuration

When you copy the v6.4 configuration to v7.0, it will preserve the TCP port numbers and Web Server document root path from your v6.4 configuration. This will mean that DataHub WebView will not be able to run until you make the following changes:

1. **TCP port numbers** - To have programs like DataHub WebView work correctly, you will need to change the TCP port numbers in your v7.0 configuration.
 - a. Run the Cogent DataHub v7.0
 - b. Open the [Tunnel/Mirror option](#) in the Properties window and change the Tunnel/Mirror Master port settings to **4502** (plain-text) and **4503** (secure) respectively.
2. **Web Server document root directory** - To start DataHub WebView or the other web demos, the Web Server root directory for v7.0 will need to be changed to point to the Cogent DataHub installation directory instead of the OPC DataHub or Cascade DataHub install directory.
 - a. Open the [Web Server option](#) in the Properties window and change the document root directory to:

C:\Program Files\Cogent\Cogent DataHub\plugin\WebServer\html

- b. The path for the Error log, Access log and SSL certificate file may also need to be changed as well.
- 3. **Web Server port number** - To avoid a conflict between the v6.4 Web Server and the v7.0 Web Server one of them needs to be using a different port number:
 - a. To change the Web Server port number, select the [Web Server option](#) in the Properties window, and change the port number from the default port 80 to another number, such as 81.

If you choose to NOT copy the v6.4 configuration

Even if you choose not to copy over the v6.4 configuration files, you need to understand that there may be a possible conflict if you want to run both v6.4 and v7.0 at the same time, on the same computer. Specifically, this is related to the DataHub Web Server that comes with each version. You cannot have two programs using the same TCP port, so you will need to configure each DataHub Web Server to use a different port number.

- 1. **Web Server port number** - To avoid a conflict between the v6.4 Web Server and the v7.0 Web Server one of them needs to be using a different port number:
 - a. To change the Web Server port number, select the [Web Server option](#) in the Properties window, and change the port number from the default port 80 to another number, such as 81.

OPC Server Changes

Since Cogent DataHub v7.0 can be run on the same computer as OPC DataHub v6.4, they must register different OPC server names. The OPC DataHub uses the PROGID `cogent.opcdatahub.1`. The Cogent DataHub uses the PROGID `cogent.datahub.1`. If you are upgrading a system from v6.4 to v7.0, any OPC clients in the system must be reconfigured to attach to the PROGID of the Cogent DataHub, namely `cogent.datahub.1`.

Copying configuration files by hand

If the configuration is not correctly copied during the first run for some reason, you can copy the OPC DataHub v6.4 configuration by hand:

1. Stop the Cogent DataHub software.
2. Remove all of the files *except for* `licenses.lic` in the Cogent DataHub configuration directory:
 - WinXP/2003 - `c:\Documents and Settings\<user>\Application Data\Cogent DataHub`
 - Windows 7 - `c:\Users\<user>\AppData\Roaming\Cogent DataHub`
3. Copy all of the files (except for `licenses.lic` if one already exists) from the v6.4 OPC DataHub configuration directory into the Cogent DataHub configuration directory.
4. Rename `Cogent DataHub\OPC DataHub.cfg` to `Cogent DataHub\Cogent DataHub.cfg`
5. Restart the Cogent DataHub v7.0.

Resetting the Cogent DataHub configuration

If you want to reset the Cogent DataHub to its default state, you can simply delete the existing configuration files. This will not affect your configuration files for v6.4:

1. Stop the Cogent DataHub software.
2. Remove all of the files *except for* `licenses.lic` in the Cogent DataHub configuration directory:
 - WinXP/2003 - `c:\Documents and Settings\<user>\Application Data\Cogent DataHub`
 - Windows 7 - `c:\Users\<user>\AppData\Roaming\Cogent DataHub`
3. Restart the Cogent DataHub v7.0.

1.6. Configuration Files

The Cogent DataHub comes with several configuration files, including a default configuration file called `default.cfg`. When the program is started for the first time, it uses `default.cfg` to create its primary configuration file, `Cogent DataHub.cfg`. This file gets edited automatically every time you make changes in the [Properties](#) window. Occasionally an experienced user might have reason to edit the `Cogent DataHub.cfg` directly, but in general it should be edited through the Properties window. When you shut down and restart the DataHub, it reads the configuration in `Cogent DataHub.cfg` to pick up where you left off.

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`. Here are the typical locations:

- For Windows XP and 2003:

`C:\Documents and Settings\User Name\Application Data\Cogent DataHub\`

- For Windows Vista, 7 and 2008:

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



If you ever need to back up the DataHub configuration, you need to copy everything in the `Cogent DataHub` directory (above). To copy the configuration from one DataHub to another PC, you need to copy everything except your license file.



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.

A user 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.

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

<code>create</code>	<code>mult</code>
<code>set</code>	<code>div</code>
<code>cset</code>	<code>lock</code>
<code>write</code>	<code>quality</code>
<code>cwrite</code>	<code>secure</code>
<code>force</code>	<code>append</code>
<code>cforce</code>	<code>dump</code>
<code>add</code>	<code>include</code>

Model-related commands allowed in config files

<code>alias</code>	<code>private_attribute</code>
<code>assembly</code>	<code>property</code>
<code>attribute</code>	<code>subassembly</code>
<code>defaultprop</code>	<code>type</code>
<code>instance</code>	

Only allowed in Windows config files

<code>execute_plugin</code>	<code>show_debug_messages</code>
<code>load_plugin</code>	<code>show_event_log</code>
<code>unload_plugin</code>	<code>show_icon</code>
<code>load_scripts</code>	<code>show_properties</code>
<code>show_data</code>	<code>show_script_log</code>

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\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 [Reference I, Cogent DataHub Command Set](#).

A small custom configuration file might look like this:

```
;;; Create some points in the default data domain
(create default:Point1)
(create default:Point2)
(create default:Point3)

;;; Assign values and confidences to the points
(set default:Point1 5 100)
(set default:Point2 67.234 100)
(set default:Point3 Hello 100)

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

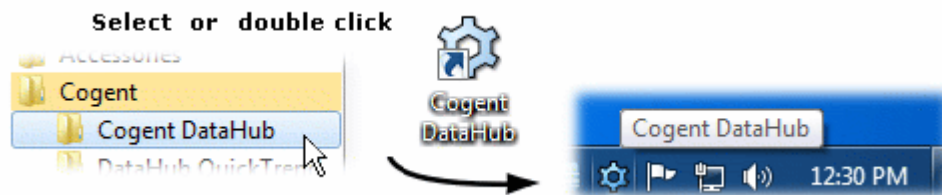
;;; Create some points in that data domain
(create NewDomain:Point1)
(create NewDomain:Point2)
(create NewDomain:Point3)


;;; Assign a value and confidence to the point
(set NewDomain:Point1 "A string" 100)
(set NewDomain:Point2 95 100)
(set NewDomain:Point3 3.1519 100)
```

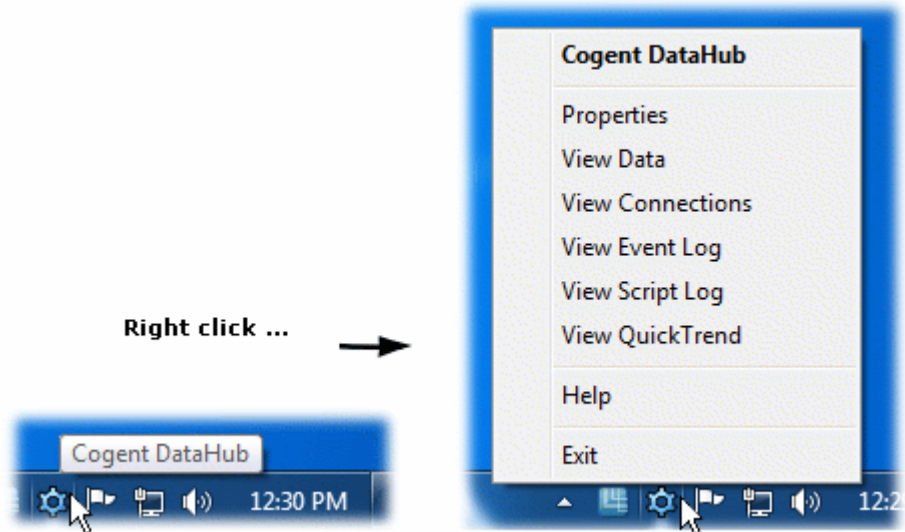
Chapter 2. Getting Started

2.1. Running the Cogent DataHub

To run the DataHub, select the program using the Windows Start menu, or double click the desktop icon.



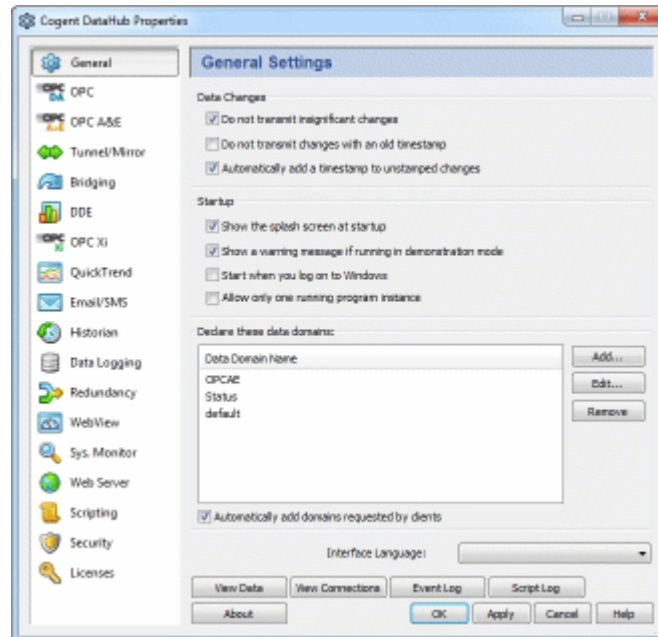
Once the DataHub is started, it runs in the background and puts a DataHub icon  in the system tray.



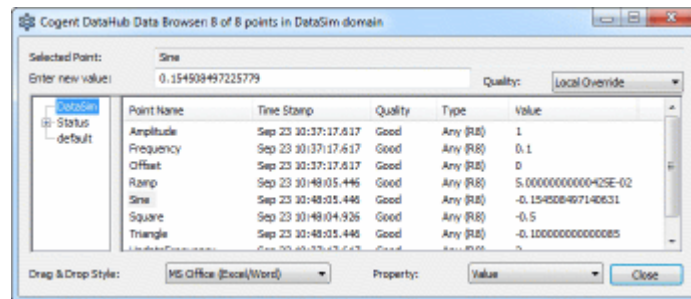
If you right-click on that icon, you will get a small pop-up menu with several options that let you open the following windows:



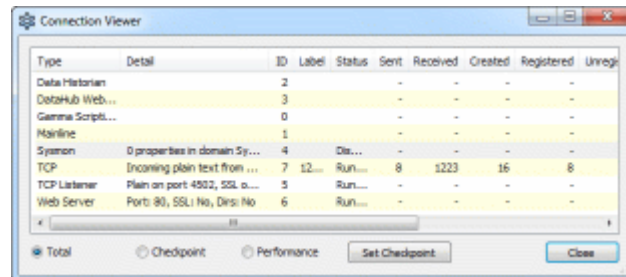
Touch-screen users: press the icon for about one second. When you release, the pop-up menu will appear.



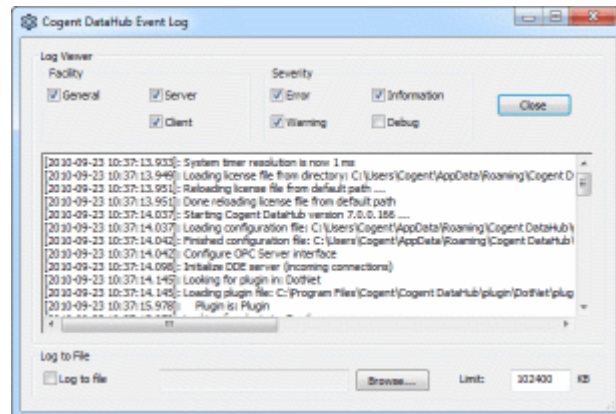
Properties window



Data Browser

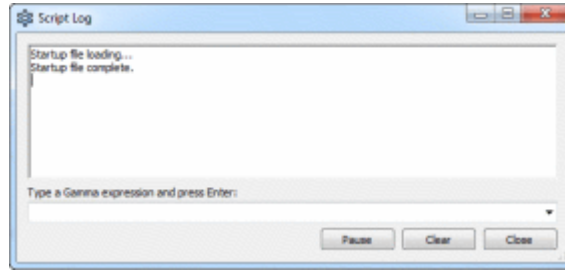


Connection Viewer

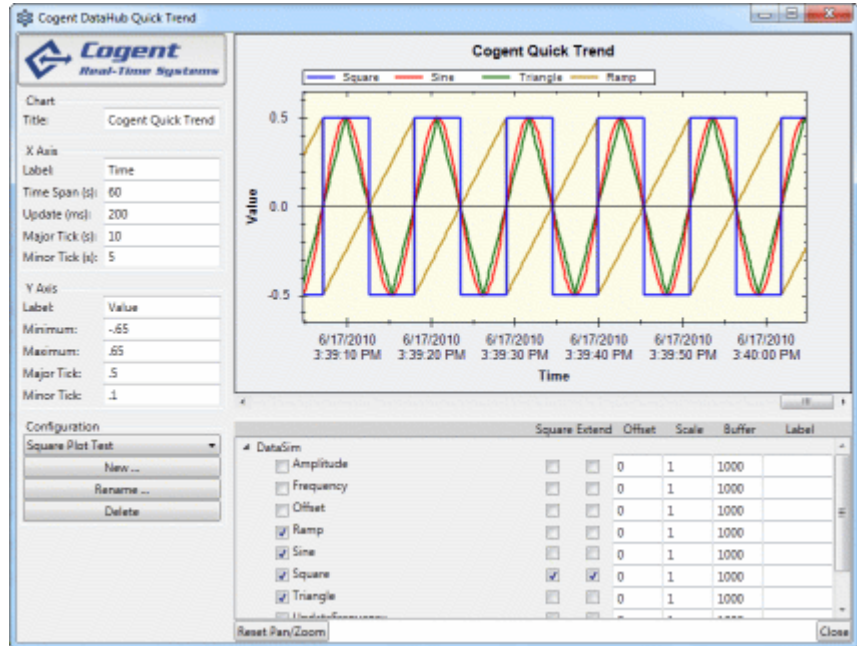


Event Log

Script Log



QuickTrend



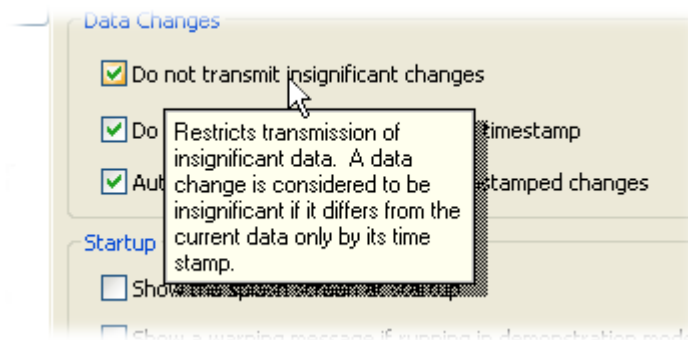
The pop-up menu also lets you [exit](#) the DataHub.




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.

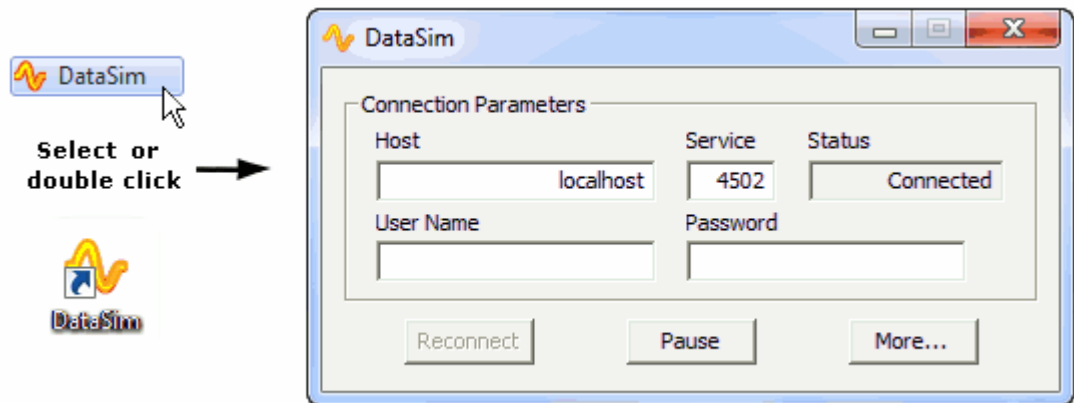



You must explicitly exit the DataHub to terminate it. Otherwise it continues to run in the background even if you close the Properties, Data Browser, and Event Log windows.

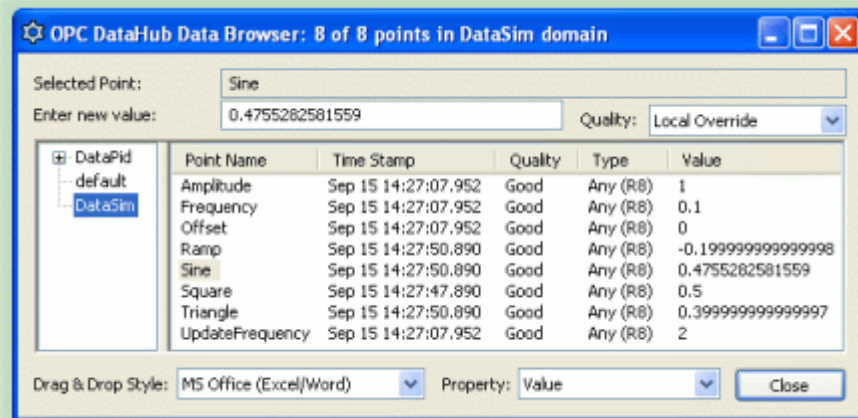
2.2. Test with simulated data

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.



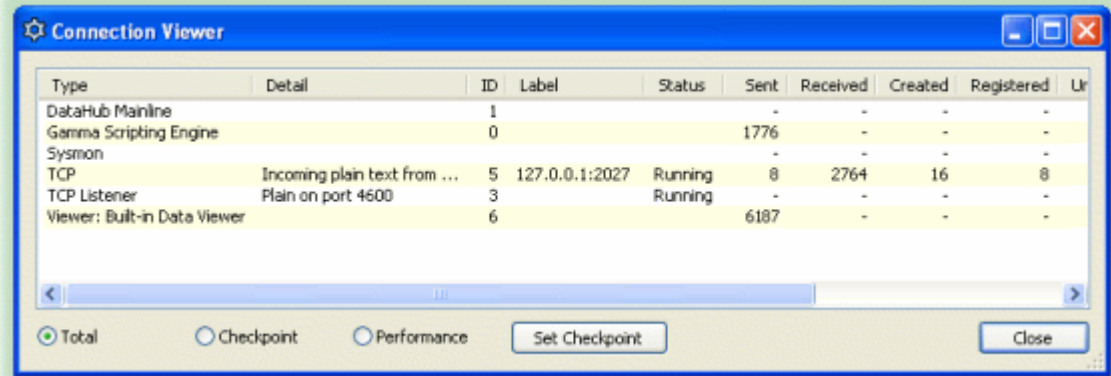
3. Open the DataHub Data Browser by right clicking on the DataHub system-tray icon  and choosing View Data from the pop-up menu.



4. Select the DataSim data domain in the left-hand pane of the window.

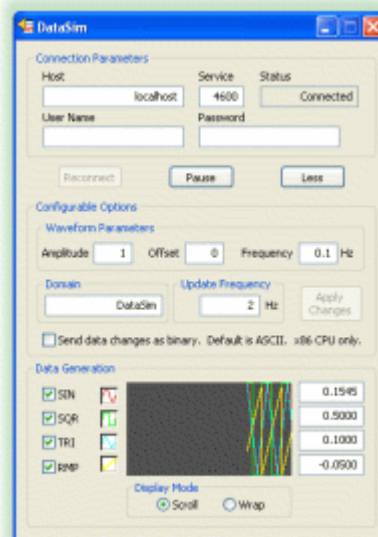
The Data Browser window should fill with simulated data updating in real time. The four updating data points are Ramp, Sine, Square, and Triangle.

5. You can also click the View Connections near the bottom of the Properties window to see your currently configured connections:



The [Connection Viewer](#) shows all active connections in the Cogent DataHub.

- 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 [Section 18.1, Data Points](#) for more information on creating and deleting points.

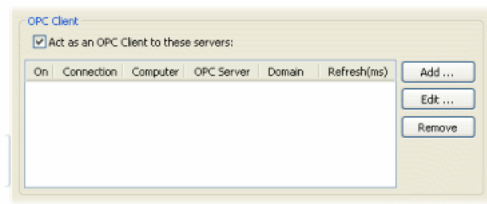
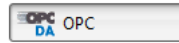
Now you are ready to start using the DataHub.

2.3. Connect to an OPC server

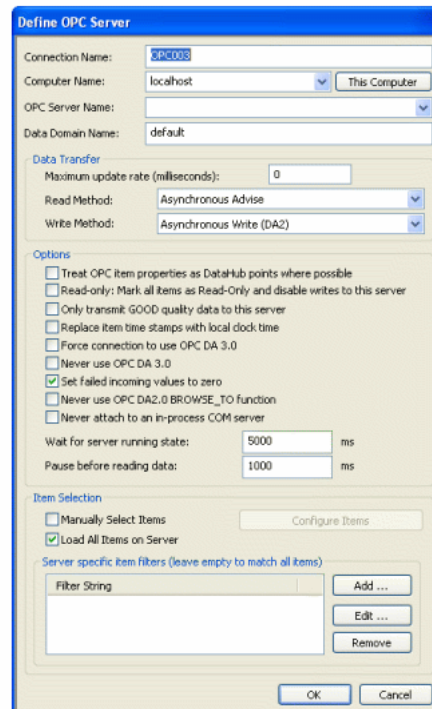
To connect to an OPC server, you need to configure the Cogent DataHub to act as an OPC client. Here's how:

- Right click on the Cogent DataHub system-tray icon and choose **Properties**.

2. In the Properties window, select OPC

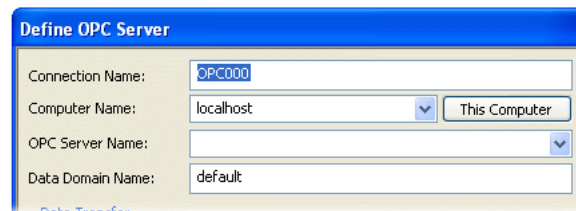


3. Check the Act as an OPC Client box. Since the DataHub can be a client to more than one OPC server, you need to specify which OPC server you are going to connect to. To add a server, click the Add button and fill in the fields in the Define OPC Server Window:



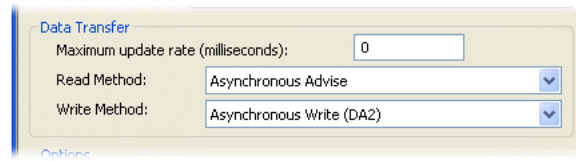
4. Type in or select the necessary information as appropriate.

- 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.

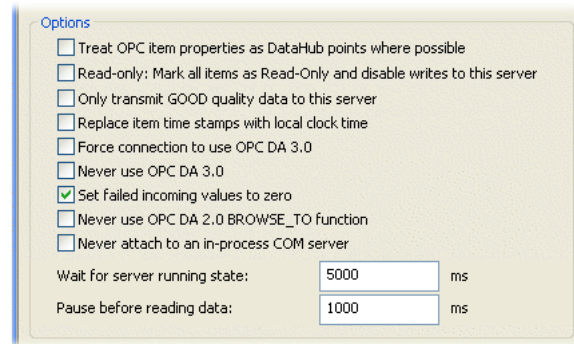
- **Data Domain Name:** type the name of the DataHub data domain in which the data points will appear.
- b. You can specify how the data is to be transferred.



- **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 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 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 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** This setting will remove 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 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 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** 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 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 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.

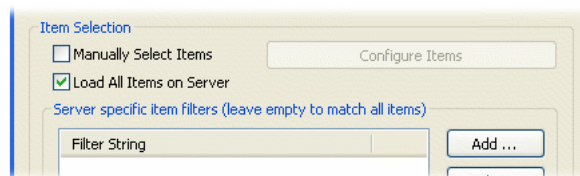
Why is this useful? An in-process server is implemented as a DLL that is loaded into the client's address space. This makes the client very dependent on the good implementation of the server. If there is a crash in an in-process server, the client also crashes. An out-of-process server is implemented 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.

- **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.

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



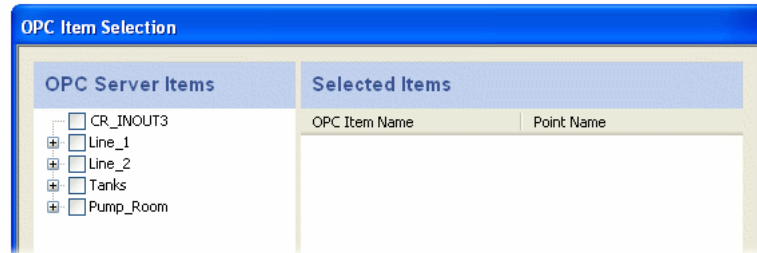
Manually Select Items



[Click here to watch a video.](#)



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:



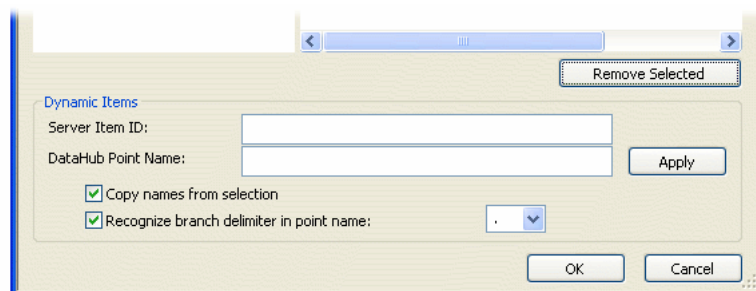
You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. Follow these guidelines for making selections:

- 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.

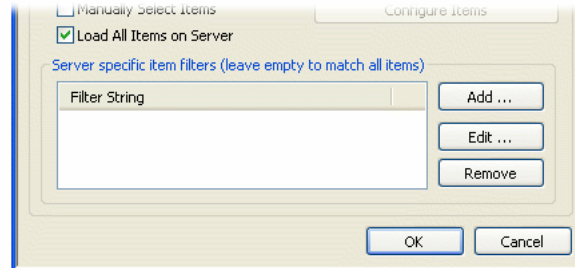


You may also configure dynamic items on the server. As you type in the **Server Item ID**, the system will fill in an identical **DataHub Point Name** for you (which you can change at any time). Press the **Enter** key or the **Apply** button to create the item. Checking the **Copy names from selection** 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.

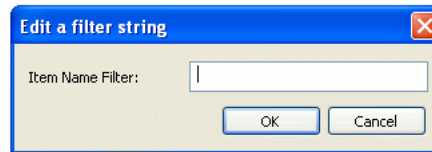
Load All Items on Server

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:



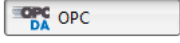
Enter a string or a pattern to match one or more item names in the OPC server. Each server has its own syntax for pattern matching, so you may have to experiment a little to get exactly the points you need. Commonly, the symbol ***** matches any number of characters, while the symbol **?** often matches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

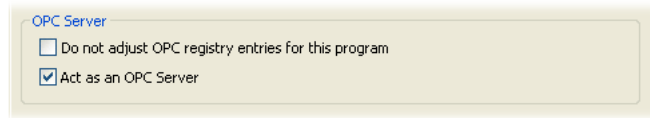
- Click the **Edit...** button to open the **Edit a filter string** window and edit an existing filter. You can do the same thing by double-clicking a filter string in the list.
 - Click the **Remove** button to remove a selected filter from the list.
5. Click the **Apply** button in the Properties Window. The DataHub should begin to act as a client to the OPC server. You can verify this using the [Data Browser](#) or the the [Connection Viewer](#). You can change these settings at any time. The Cogent DataHub will reconnect and apply the changes when you click the **Apply** button in the Properties Window.

2.4. Connect from an OPC client

When you start an OPC client it should immediately connect to the Cogent DataHub, because the DataHub is preconfigured to act as an OPC 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** .



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.OPCDataHub` or `Cogent.OPCDataHub.1`.

The Do not adjust OPC registry entries for this program option tells the Cogent DataHub not to alter its registry settings. This is useful if you want to use the Cogent 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.

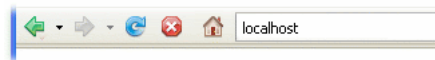
These two boxes work together, because turning the OPC server behavior on or off necessarily makes changes to the registry. Here is how you can change OPC server behavior when you also need to maintain registry settings:

- a. Uncheck Do not adjust OPC registry entries for this program. This will make the Act as an OPC Server checkbox visible.
 - b. Check or uncheck the Act as an OPC Server as needed, and click Apply.
 - c. Check Do not adjust OPC registry entries for this program and click Apply.
4. Click Apply button at the bottom of the Properties window to apply the change. You can view connections with the [Connection Viewer](#).

2.5. Test the Web Server

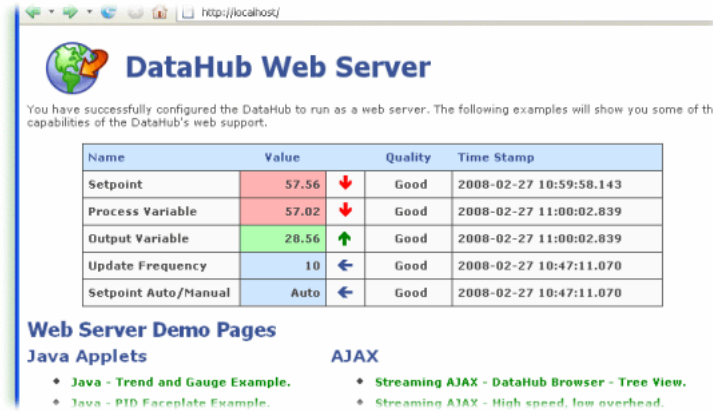
To test Cogent DataHub's web server, follow these steps:

1. Ensure that the DataHub is running, and that the DataHub Web Server is enabled.
2. Type the IP address of the DataHub into the Address field at the top of your web browser.



If you are running the DataHub and web browser on the same machine, type `localhost`. Otherwise, type the IP address or computer name of the computer running the DataHub.

3. The welcome page should appear. You can follow the links to see the various demos. The live data in the demos comes from DataPid. If you don't see data updating, you'll need to start DataPid.



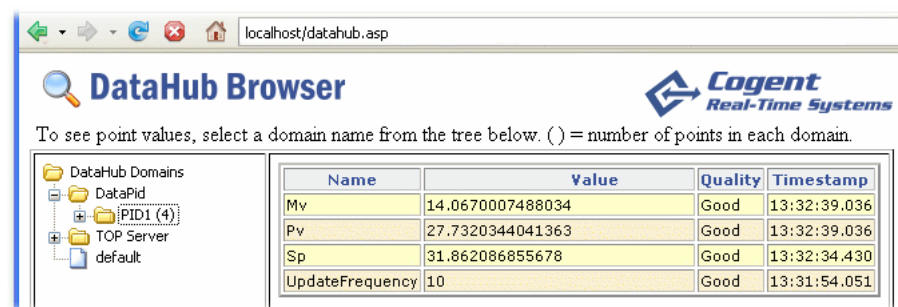
If the test doesn't work, please refer to [Section 8.2, Configuring the Web Server](#) for more information.

2.6. View Your Data on the Web

You can view all the data points currently in the DataHub using the Web Data Browser.

1. Ensure that the DataHub is running and connected to your data sources, and that the DataHub Web Server is enabled.
2. If you are running your web browser on the same machine as your DataHub, type `localhost/datahub.asp` into the Address field at the top of your browser. If you are running your web browser on a different machine from the DataHub, type the IP address or computer name of the computer running the DataHub, instead of `localhost`.

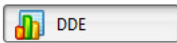
The Web Data Browser page should display:

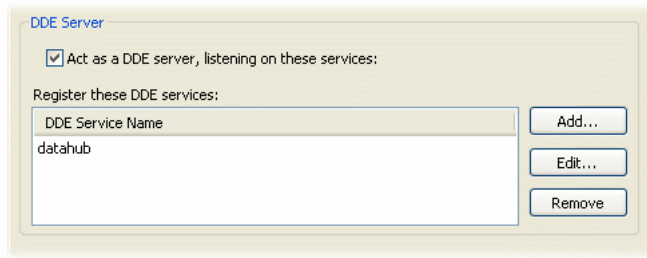


The DataHub domains and data hierarchy are displayed in the left pane, and live updates of the point values for the selected part of the hierarchy are shown the right pane. You can modify the header for this page by editing this file: `C:\Program Files\Cogent\Cogent DataHub\Plugin\Webserver\html\domaintreeheader.asp`.

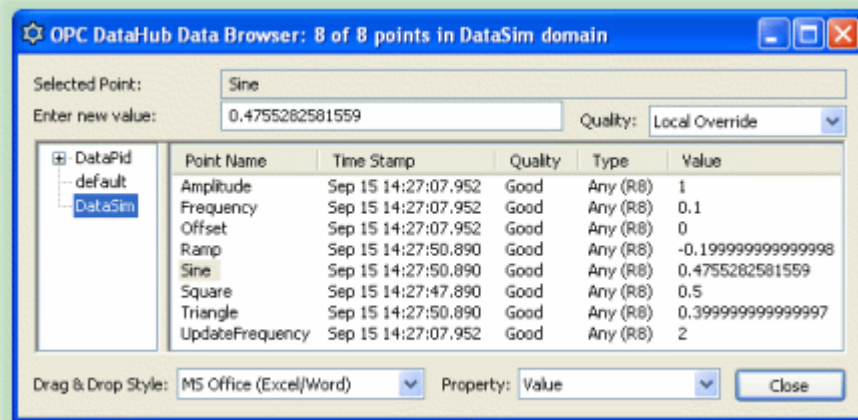
2.7. Test with Excel

To test the Cogent DataHub by putting live data into Excel, follow these steps:

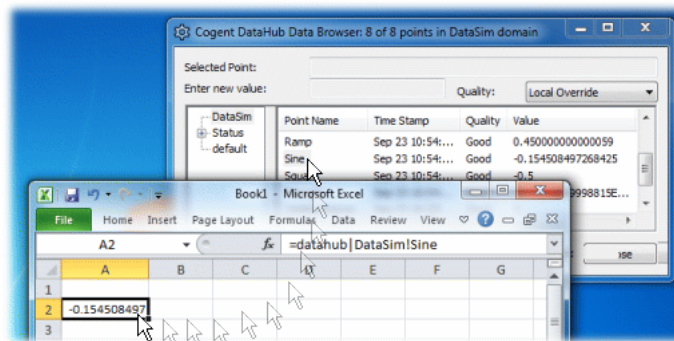
1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.
2. In the Properties window, select **DDE** .



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. Start DataSim if it isn't already running, to get live data in 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.

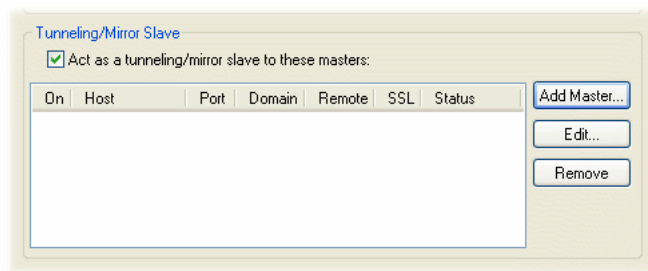


You can drag and drop timestamps and other attributes of a point using the Property dropdown list. Please refer to [Drag and Drop Style and Property](#) in the [Data Browser](#) section for more details.

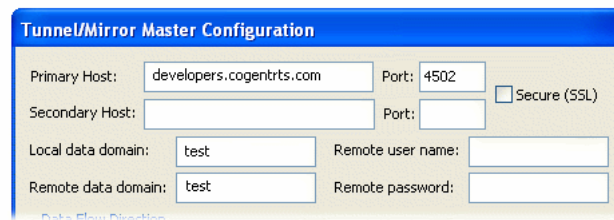
2.8. Connect to remote data

Cogent has a DataHub running that makes live test data available over the Internet. To set up 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.



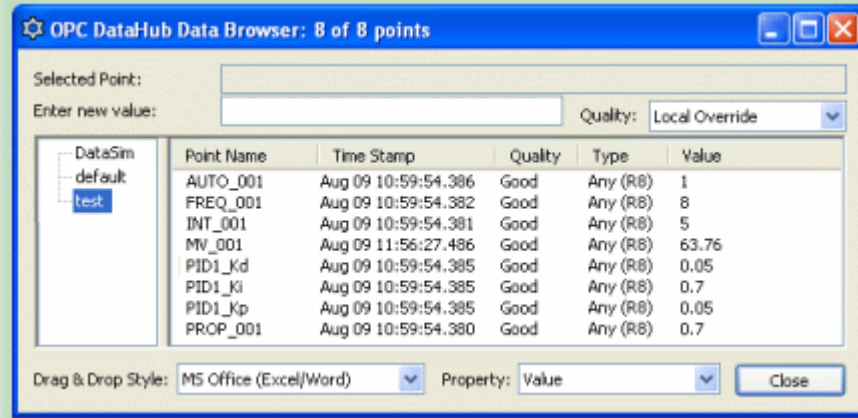
4. Click the Add Master... button to open the Tunnel/Mirror Master Configuration window:



5. Fill in these entry fields at the top as follows:
 - **Primary Host:** `developers.cogentrts.com`
 - **Local data domain:** `test`
 - **Remote data domain:** `test`

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 `test` 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 [Section 20.4, Tunnel/Mirror](#).

Chapter 3. OPC Tunnelling

3.1. 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 offers OPC tunnelling to avoid the problems of DCOM configuration.



Tunnelling is the same as *mirroring*, as described in [Section 6.4, *Networking Excel*](#). Tunnelling can be also used to connect to Cascade DataHub running in Linux. Please refer to the Mirroring Data section of the Cascade DataHub for Linux and QNX manual for details.

Using the Cogent DataHub for OPC tunnelling means:

- Connect equally easily across a LAN or WAN.
- No DCOM—no timeouts or configuration problems.
- Complete and secure data access.
- Simple set up.

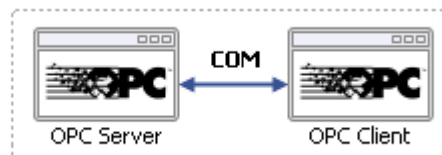
Setting up the Cogent DataHub for OPC tunnelling is a simple, 3-step configuration that can be done in a few minutes. All you need to do is:

1. [Configure the DataHub on the OPC server machine.](#)
2. [Configure the DataHub on the OPC client machine.](#)
3. [Start the OPC client.](#)

Why use tunnelling for OPC DA?

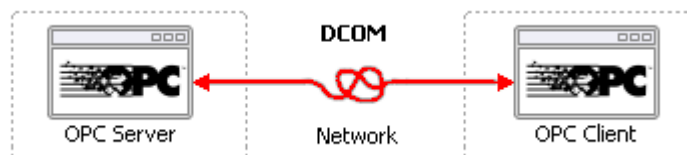
Most people who attempt to network OPC DA servers and clients experience a number of problems. Networking is just not OPC's strength. OPC was originally based on COM (component object model), which runs on a single computer.

Easy:
OPC on a
single node
using COM.



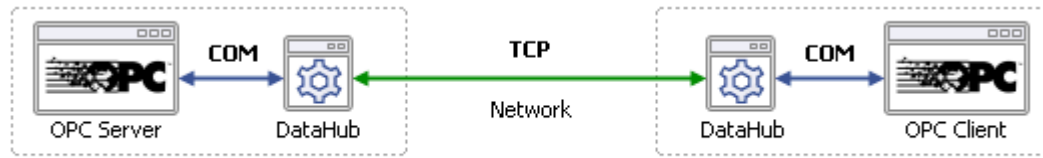
As long as the OPC server and OPC client are on one computer, setting up the connection between them is easy. Difficulties arise, however, when the OPC server and client are on different computers, and need to be networked.

Difficult:
OPC over
a network
using DCOM.



To communicate over a network, OPC 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.



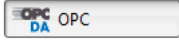
Solution: Eliminate DCOM. Use COM at each end and TCP over the network.

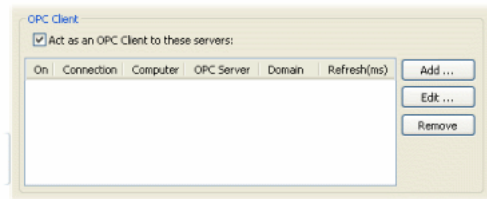
The Cogent DataHub provides a COM (OPC) interface for the OPC client and server, and uses TCP across the network. This is known as *tunnelling*. Each connected OPC server or client sees the other as a local OPC connection. They are unaware that their messages are being converted to TCP along the way. Tunnelling works equally well across a LAN or WAN, and it is even possible to tunnel through firewalls.

3.2. Configuring the DataHub for the server

Configure the DataHub as an OPC client

The DataHub on the OPC 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 .



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 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.

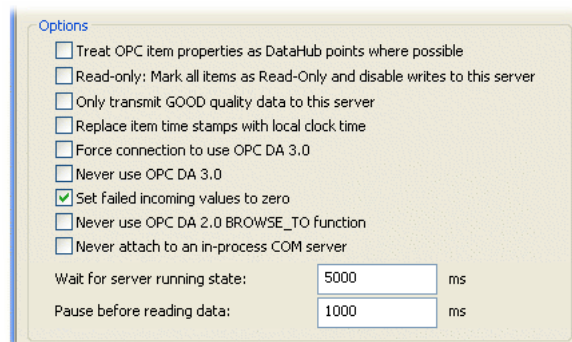
- **Maximum update rate (milliseconds):** Enter the maximum rate you wish the data to be updated. This is useful for slowing down the rate of incoming data. The default is

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 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 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 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** This setting will remove 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 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 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** 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 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 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

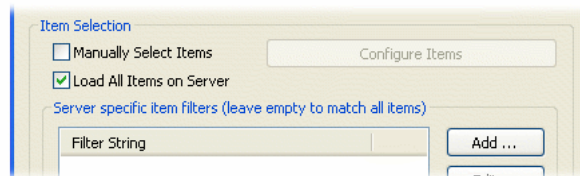
of the server. If there is a crash in an in-process server, the client also crashes. An out-of-process server is implemented as a separate executable. The client communicates with an out-of-process server using the inter-process communication mechanisms in DCOM. In theory an in-process server will be 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.

- **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.

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



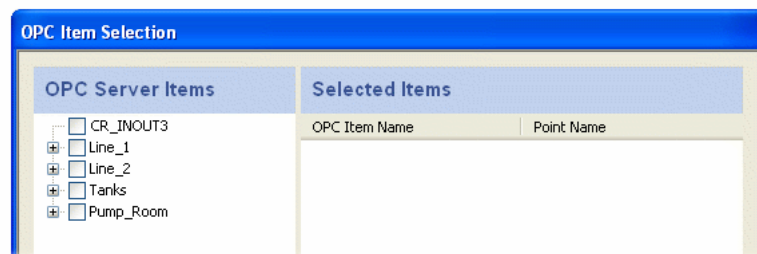
Manually Select Items



[Click here to watch a video.](#)



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:



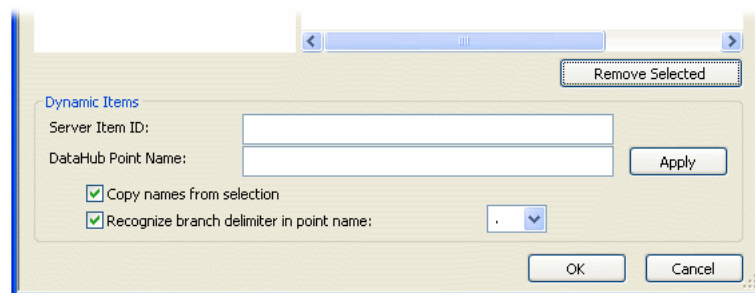
You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. Follow these guidelines for making selections:

- 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.

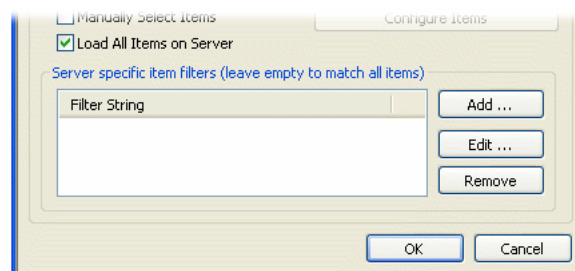
- Selecting a server item does not automatically add any of its child items. Each child item must be added separately. To view child items, click the + sign in front of the item. If an item has one or more children that have been selected, the item name(s) will appear in bold.
- To delete selected items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

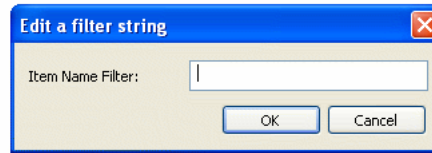
Load All Items on Server

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:




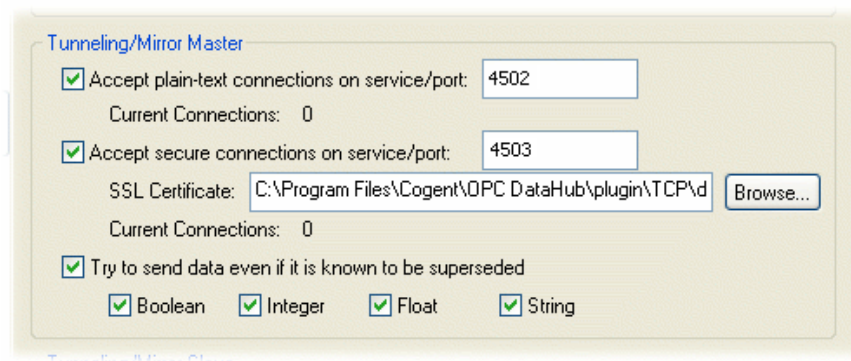
Enter a string or a pattern to match one or more item names in the OPC server. Each server has its own syntax for pattern matching, so you may have to experiment a little to get exactly the points you need. Commonly, the symbol `*` matches any number of characters, while the symbol `?` often matches 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.
- Click the **Apply** button in the Properties Window. The DataHub should begin to act as a client to the OPC server. You can verify this using the [Data Browser](#) or the [Connection Viewer](#). You can change these 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.

- Right click on the Cogent DataHub system-tray icon and choose **Properties**.
- 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 [Section 20.4, 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. Click OK to close the Properties window.


The OPC server machine side of the tunnelling connection is now ready, and you can move to the OPC client machine.

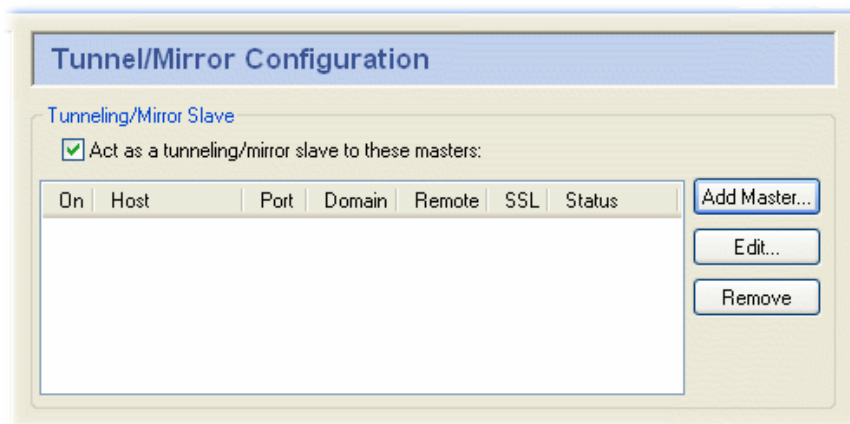
3.3. Configuring the Cogent DataHub for the client

Now you need to set up the Cogent DataHub on this machine to tunnel across to the DataHub on the OPC server machine.

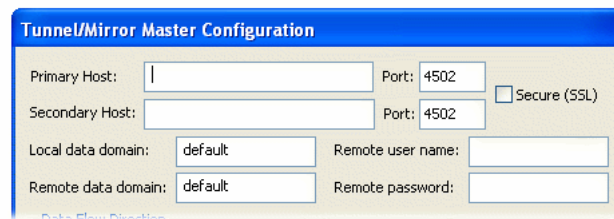
Configure the Cogent DataHub as tunnelling slave

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 OPC client side act as the tunnelling slave, while the DataHub on the OPC 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**.



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 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.

- **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.



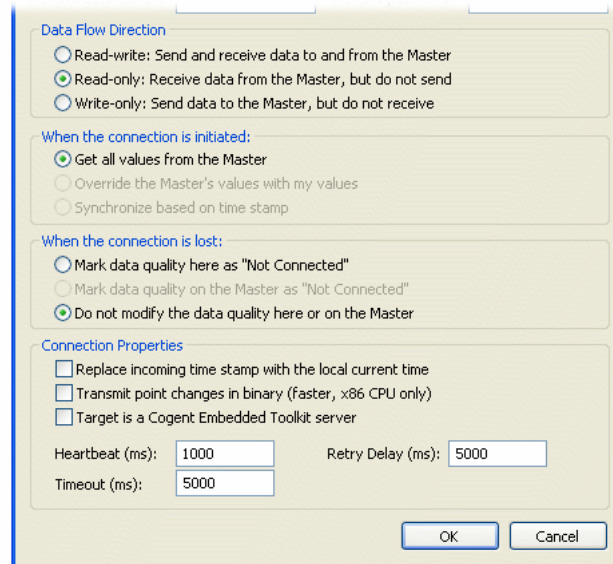
Unless you have a good reason for making these different, we recommend using the same data domain name on both DataHubs for the sake of simplicity.



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

- **Primary Host:** `developers.cogentrts.com`
- **Port:** 4502
- **Local data domain:** `test`
- **Remote data domain:** `test`

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 [Section 19.1, Binary Mode Tunnel/Mirror \(TCP\) Connections](#).

- **Target is a Cogent 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.



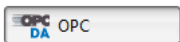
To optimize this setting for slow networks, please refer to [Section 19.2, Tunnel/Mirror \(TCP\) Connections for Slow Networks](#).

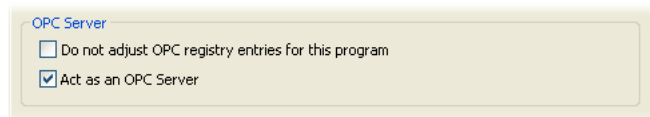
- **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

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 to act as a server to the OPC client

Finally, we suggest that you ensure that the Cogent DataHub on the OPC 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 .



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.OPCDataHub` or `Cogent.OPCDataHub.1`.

The Do not adjust OPC registry entries for this program option tells the Cogent DataHub not to alter its registry settings. This is useful if you want to use the Cogent 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 server behavior when you also need to maintain registry settings:

- a. Uncheck Do not adjust OPC registry entries for this program. This will make the Act as an OPC Server checkbox visible.
 - b. Check or uncheck the Act as an OPC Server as needed, and click Apply.
 - c. Check Do not adjust OPC registry entries for this program and click Apply.
4. 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 client, connect to the Cogent DataHub, and access your data.

3.4. Testing the connection

You can test your tunnelling connection like this:

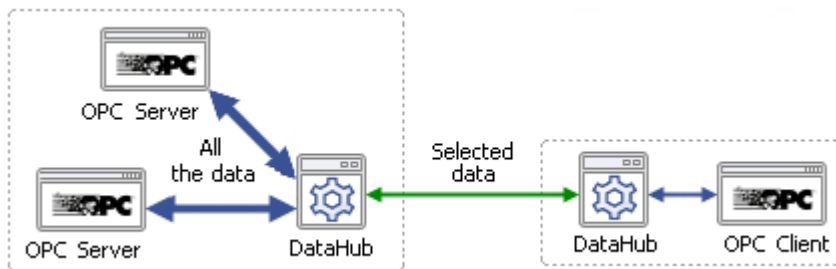
1. Ensure that you have correctly set up the [OPC server machine](#) and the [OPC client machine](#).
2. Start the Cogent DataHub on the OPC server machine if it is not running already. It should start up the OPC server on that machine.
3. Start the OPC client on the OPC client machine. It should start the Cogent 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 Cogent DataHub setup on both machines.
- The functioning of the OPC server and client.
- The physical network connection.

3.5. Tunnelling part of the data set

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 [OPC bridging](#). When you [configure bridges](#), just make sure to create target points in a new, separate data domain. For more information on OPC bridging, please refer to [Chapter 5, OPC Bridging](#).

3.6. Extended applications

Tunnelling by itself greatly enhances the usefulness of OPC. However, you can get even more benefit out of the Cogent DataHub by combining tunnelling with OPC bridging and/or aggregation.

3.6.1. Tunnelling and Bridging

[OPC bridging](#) means linking data from one OPC server to another OPC server, usually on a single machine. However, you can bridge two OPC servers over a network using a tunnelling connection:

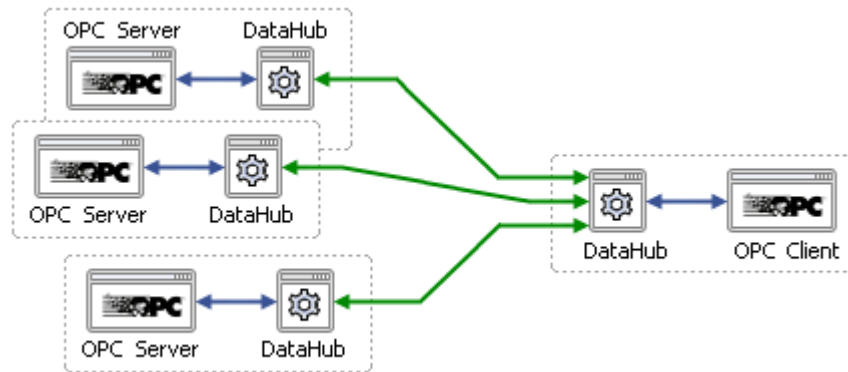


This scenario involves setting up the DataHubs on both machines to **act as OPC clients** to the respective OPC servers. The DataHubs then interface with each other over a TCP tunnelling connection. Configure the DataHub on the machine with the most uptime to be the **tunnelling master** and the other DataHub to be the **tunnelling slave**. We recommend that all **bridges be configured** on just one of the DataHubs.

3.6.2. Tunnelling and Aggregation

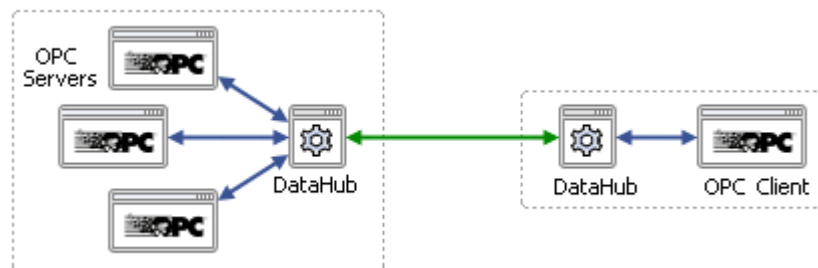
OPC aggregation means combining data from multiple OPC servers into a single server. Here are a few ways that tunnelling can be combined with aggregation:

1. **Aggregation from remote servers** uses tunnelling to bring the data from several OPC servers on different machines into one client.



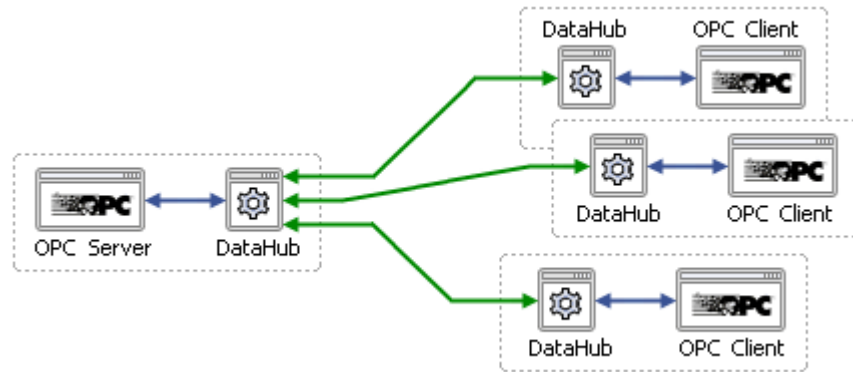
This scenario involves setting up three **OPC server machines** and one **OPC client machine** for tunnelling. **Aggregation** takes place on the OPC client machine.

2. **Remotely connecting to several servers** does tunnelling to connect a group of servers on a single machine to a remote client.



This scenario requires **configuring the DataHub** for each of the three different OPC servers, and setting up the **OPC client machine** for tunnelling. **Aggregation** takes place on the OPC server machine.

3. **Remotely connecting many clients** uses 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.

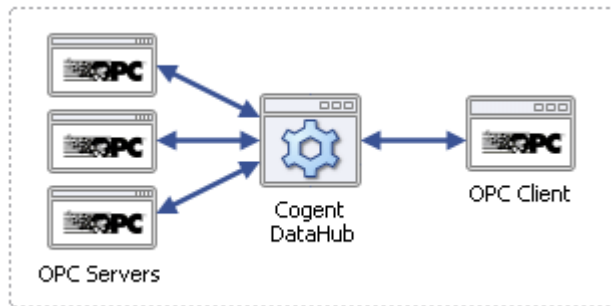
4. Many other combinations can be made. These are just a few to get you started.

For more information about using the Cogent DataHub to aggregate clients or servers, please refer to [Chapter 4, OPC Aggregation](#).

Chapter 4. OPC Aggregation

4.1. Introduction

OPC *aggregation* means bringing together the data from several OPC servers into one common access point, a single server. [Configuring the DataHub](#) for aggregation is simply a matter of adding servers.



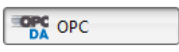
The Cogent DataHub can aggregate any number of servers, and act as a server to any number of clients. Servers can be added or removed dynamically, during run-time.

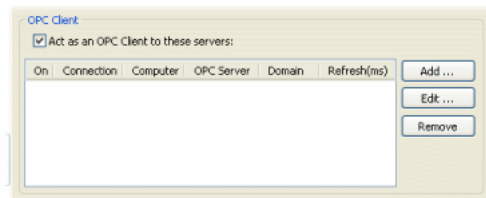
4.2. Configuring the DataHub

Configure the DataHub to act as a client to OPC servers

To aggregate data from OPC servers, you need to set up the DataHub to act as a client to each server. Here's how:

1. Right click on the Cogent DataHub system-tray icon and choose **Properties**.

2. In the Properties window, select **OPC** .



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 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.

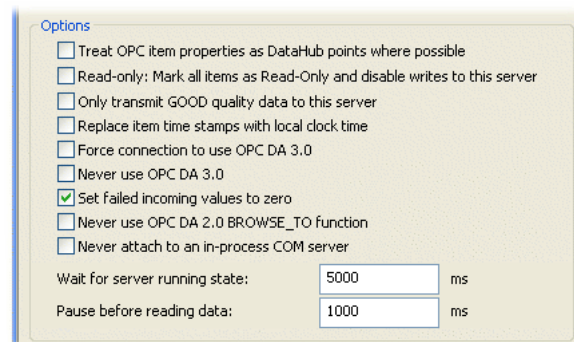
- **Maximum update rate (milliseconds):** Enter the maximum rate you wish the data to be updated. This is useful for slowing down the rate of incoming data. The default is

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 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 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 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** This setting will remove 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 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 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** 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 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 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

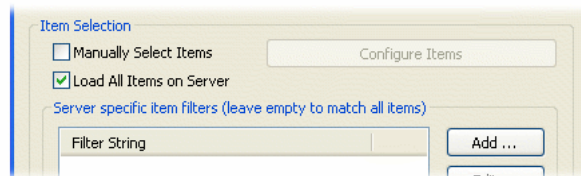
of the server. If there is a crash in an in-process server, the client also crashes. An out-of-process server is implemented as a separate executable. The client communicates with an out-of-process server using the inter-process communication mechanisms in DCOM. In theory an in-process server will be 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.

- **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.

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



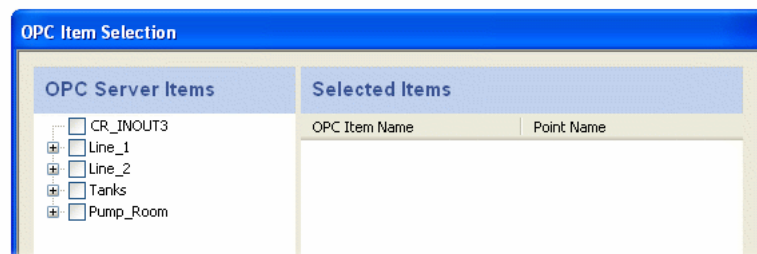
Manually Select Items



[Click here to watch a video.](#)



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:



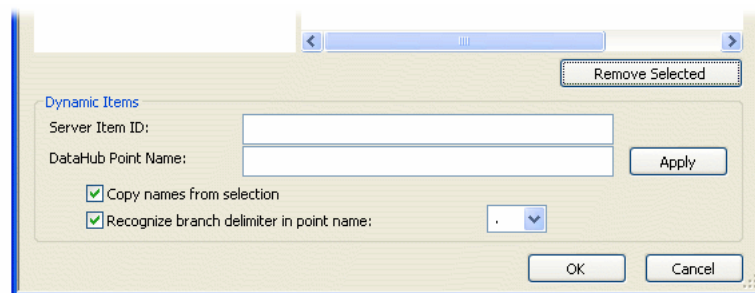
You can browse through the tree in the left pane, selecting points as you go. The selections will appear in the right pane. Follow these guidelines for making selections:

- 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.

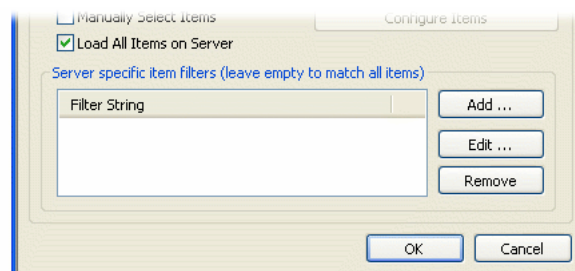
- Selecting a server item does not automatically add any of its child items. Each child item must be added separately. To view child items, click the + sign in front of the item. If an item has one or more children that have been selected, the item name(s) will appear in bold.
- To delete selected items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

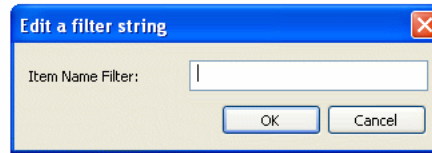
Load All Items on Server

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.

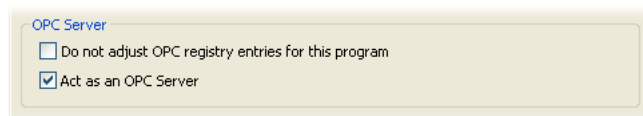
- Click the **Edit...** button to open the **Edit a filter string** window and edit an existing filter. You can do the same thing by double-clicking a filter string in the list.
 - Click the **Remove** button to remove a selected filter from the list.
- Click the **Apply** button in the Properties Window. The DataHub should begin to act as a client to the OPC server. You can verify this using the [Data Browser](#) or the [Connection Viewer](#). You can change these settings at any time. The Cogent DataHub will reconnect and apply the changes when you click the **Apply** button in the Properties Window.

To add another OPC server, simply repeat the steps above.

Configure the DataHub to act as an OPC server

To get the aggregated data into one or more OPC clients, you need to set up the Cogent DataHub to act as a server, as follows:

- Right click on the DataHub system-tray icon and choose **Properties**.
- In the Properties window, select **OPC**.



- 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.OPCDataHub` or `Cogent.OPCDataHub.1`.

The **Do not adjust OPC registry entries for this program** option tells the Cogent DataHub not to alter its registry settings. This is useful if you want to use the Cogent 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 server behavior when you also need to maintain registry settings:

- a. Uncheck Do not adjust OPC registry entries for this program. This will make the Act as an OPC Server checkbox visible.
 - b. Check or uncheck the Act as an OPC Server as needed, and click Apply.
 - c. Check Do not adjust OPC registry entries for this program and click Apply.
4. Click Apply button at the bottom of the Properties window to apply the change. You can view connections with the [Connection Viewer](#).

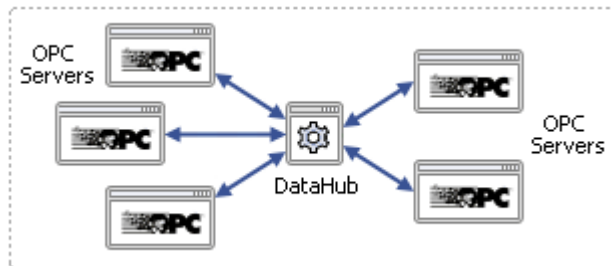
Once you have configured your OPC clients to work with the DataHub, you should be able to access the data from all the aggregated OPC servers in any OPC client.

4.3. Extended applications

Aggregation gives you more access to your OPC data, but why stop there? With the Cogent DataHub you can use OPC tunnelling and/or bridging to put the aggregated data where you need it.

4.3.1. Aggregation and Bridging

[OPC bridging](#) means linking data from one OPC server to another OPC server. With the Cogent DataHub you can combine aggregation with bridging to bridge multiple OPC servers simultaneously.

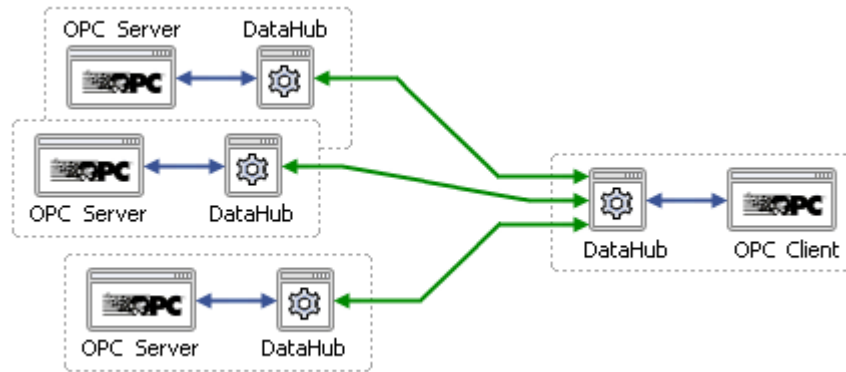


This scenario involves [aggregating](#) five different OPC servers and then [bridging](#) the data between them.

4.3.2. Aggregation and Tunnelling

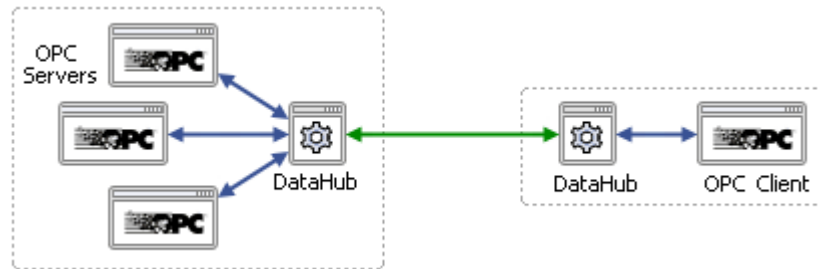
[OPC tunnelling](#) means sending OPC data across a network without the hassles of DCOM. Here are a few ways that aggregation can be combined with tunnelling:

1. **Aggregating remote servers** uses tunnelling to bring the data from several OPC servers on different machines into one client.



This scenario involves setting up three [OPC server machines](#) and one [OPC client machine](#) for tunnelling. [Aggregation](#) takes place on the OPC client machine.

2. **Remotely connecting to several servers** aggregates a group of servers on a single machine and uses tunnelling to connect to a remote client.



This scenario requires [aggregating](#) three different OPC servers and configuring the DataHub as a [tunnelling master](#), then setting up the [OPC client machine](#) for tunnelling. [Aggregation](#) takes place on the OPC server machine.

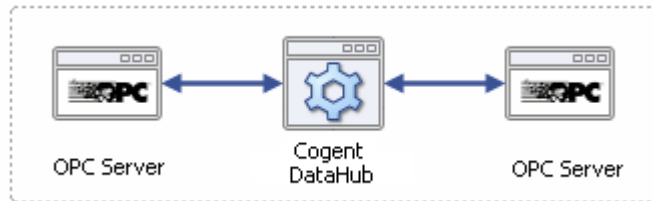
3. Many other combinations can be made. These are just a few to get you started.

For more information about using the Cogent DataHub to tunnel through firewalls and connect OPC across a network, please refer to [Chapter 3, OPC Tunnelling](#).

Chapter 5. OPC Bridging

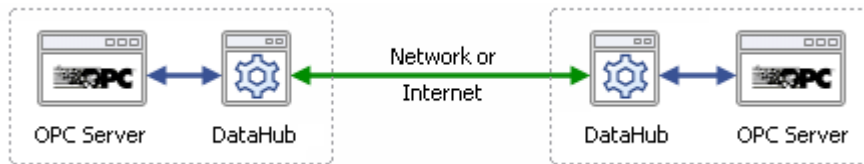
5.1. Introduction

OPC *bridging* means connecting two OPC servers together so they can access each other's data.

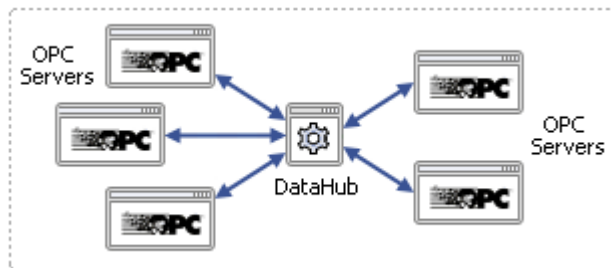


In addition to bridging two OPC servers on a single computer, the Cogent DataHub offers advanced bridging capabilities that let you:

- Bridge OPC servers over a network connection, by [tunnelling](#).



- Bridge between any number of OPC 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](#).
- Bridge between [Excel and OPC](#).

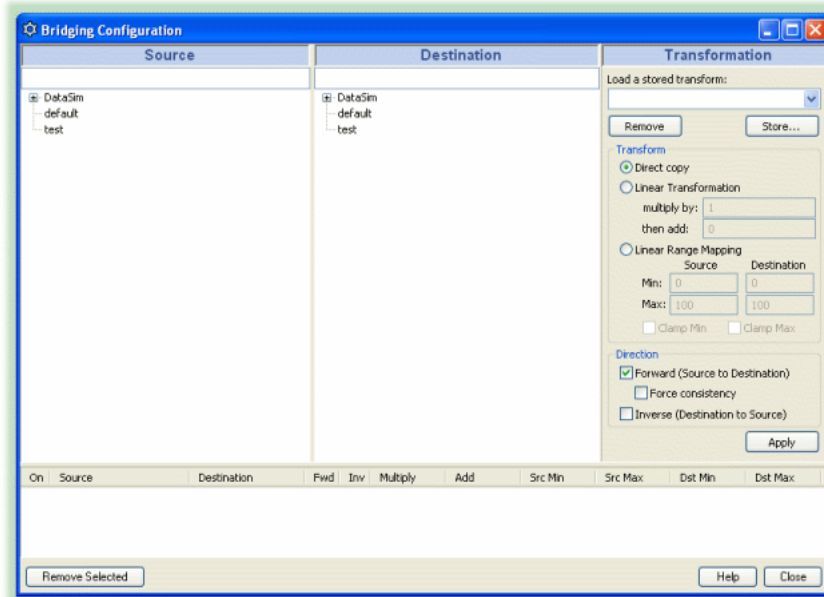
You can [configure the bridges](#) you need using the Bridging option in the Properties Window.

5.2. 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.

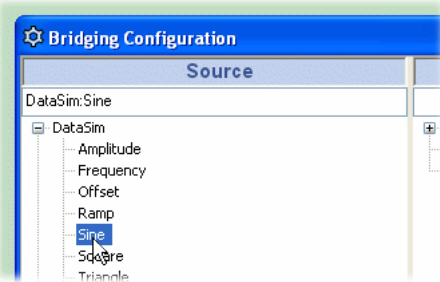
5.2.1. Point-to-point configuration

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.



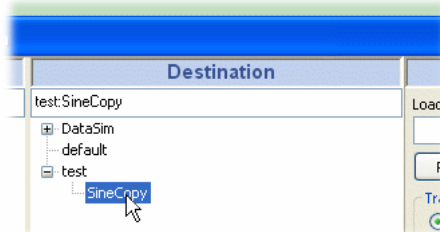
The three top panes in this window correspond to the three basic steps in making the configuration: specify a source, a destination, and any desired transformations. The horizontal pane across the bottom shows the bridges that exist on the system.

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.

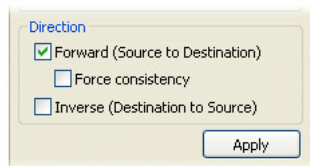
- Specify direct copy or transformation.



The **Transform** dialog box contains three radio button options: **Direct copy** (selected), **Linear Transformation**, and **Linear Range Mapping**. Under **Linear Transformation**, there are input fields for **multiply by:** (value: 1) and **then add:** (value: 0). Under **Linear Range Mapping**, there are two columns: **Source** and **Destination**. Each column has **Min:** and **Max:** input fields, both currently set to 0 and 100 respectively. At the bottom, there are two checkboxes: **Clamp Min** and **Clamp Max**, both of which are unchecked.

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 [Section 5.2.2, Making transformations](#) below.

- Determine which direction you want the bridge to apply.



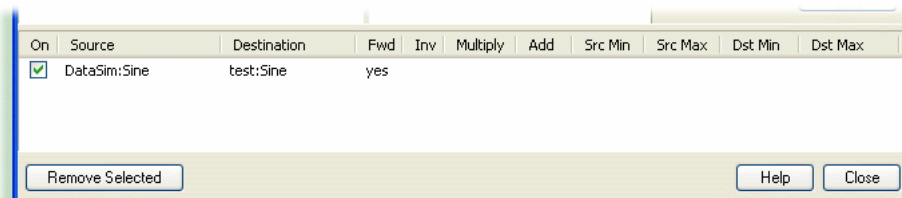
The **Direction** dialog box has two checked options: **Forward (Source to Destination)** and **Inverse (Destination to Source)**. There is an unchecked checkbox for **Force consistency**. An **Apply** button is located at the bottom right.

- 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.
- Click the **Apply** button to create and activate the bridge. The DataHub will create the bridge and update the bridged points immediately.
 - In the bottom panel you can see all the bridges that exist in the system, and the significant information about them.



On	Source	Destination	Fwd	Inv	Multiply	Add	Src Min	Src Max	Dst Min	Dst Max
<input checked="" type="checkbox"/>	DataSim:Sine	test:Sine	yes							

Buttons: Remove Selected, Help, Close

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.

5.2.2. 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:

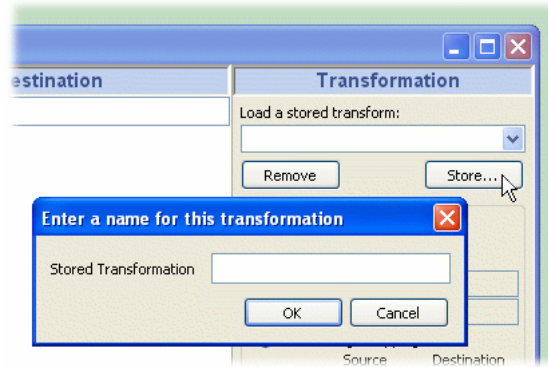
$$\text{Celsius} = (\text{Fahrenheit} - 32) / 1.8$$

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**.

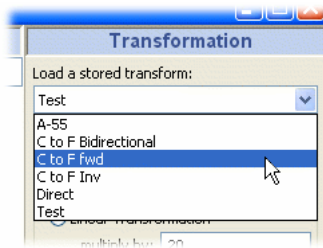
When you use linear range mapping, you can limit the transformed value to the maximum and minimum by checking the **Clamp** boxes. The clamps get applied to the point being changed, ie. to the destination point for forward direction, to the source point for inverse direction, and to both points for bidirectional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

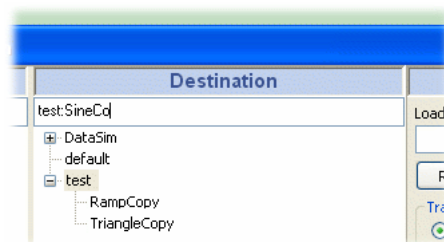


5.3. Creating New Points

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.



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.



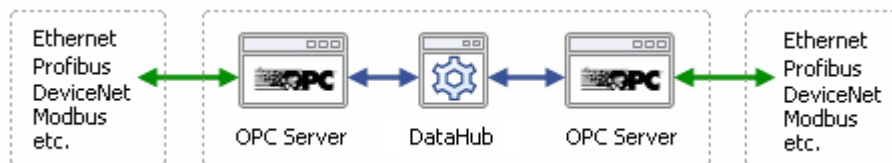
You might want to create special data domains for holding sets of destination points. For more information about data domains, please refer to [Section 18.4.1, Data Domains](#).

5.4. Bridging Scenarios

Here are some of the most commonly used bridging scenarios.

5.4.1. Bridging Local Servers

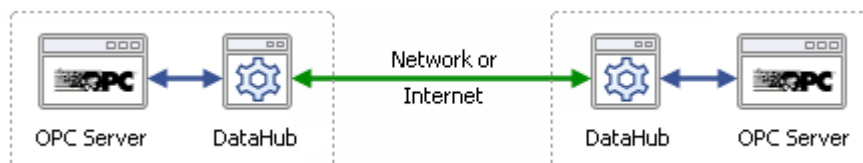
The most common scenario for OPC bridging is connecting OPC servers on a single machine. This can be used to create connections over various fieldbus protocols. Any fieldbus connected to an OPC server can be bridged to any other.



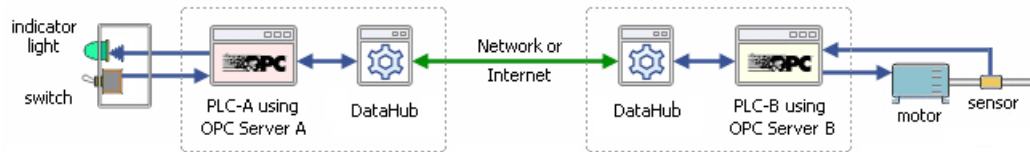
The bridge is [configured](#) as a direct link, or it might incorporate [linear transformations](#). It can have a forward or inverse direction, or be bidirectional.

5.4.2. 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:

Point	Function
A.switch	- Changes value to 1 when switch is on, 0 when switch is off.
A.indicator	- Turns on light when value is 1, switches off light when value is 0.
B.motor	- Starts motor when value is 1, switches off motor when value is 0.
B.sensor	- Changes value to 1 when shaft is turning is on, 0 when shaft is not turning.

then bridge them in the DataHub like this:

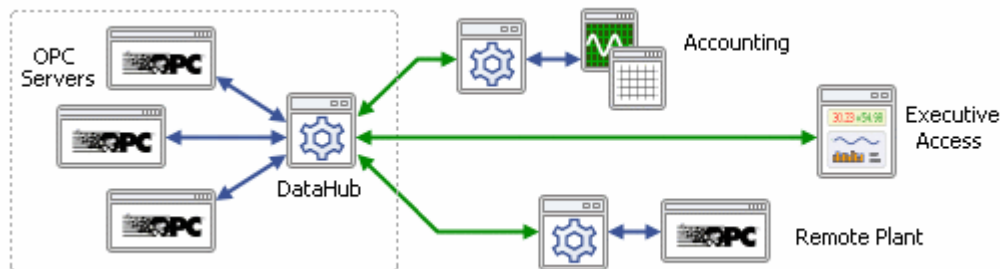
On	Source	Destination	Fwd	Inv	Multi
<input checked="" type="checkbox"/>	A:switch	B:motor	yes		
<input checked="" type="checkbox"/>	A:indicator	B:sensor		yes	

As when bridging local servers, bridging remote servers can also bridge different fieldbus protocols.

5.4.3. 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:

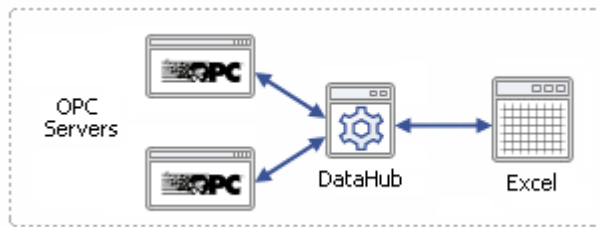
- It simplifies configuration and use.
- It reduces bandwidth across the network.



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.

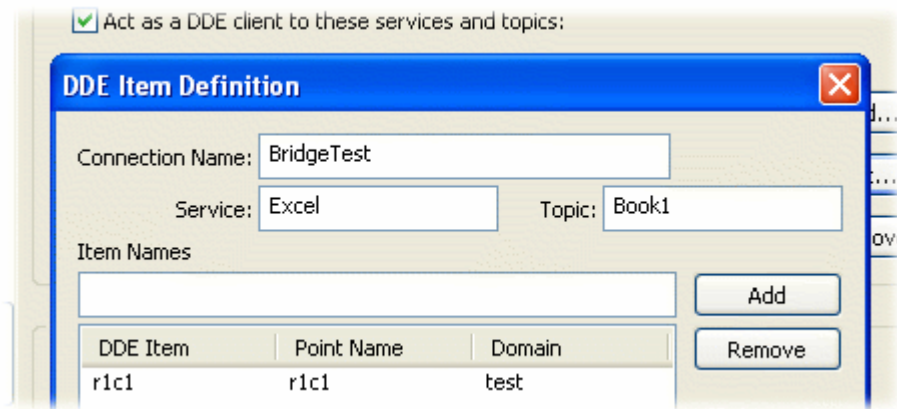
5.4.4. Bridging to Excel

The Cogent DataHub makes it easy bridge data from an Excel spreadsheet to an OPC server.

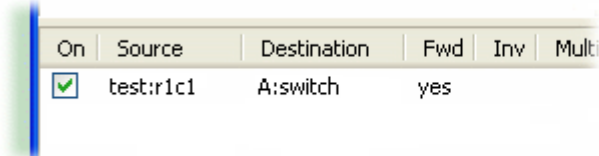


Simply [connect the DataHub to Excel](#), then [configure the bridges](#). To get Excel data into an OPC server without changing any names in Excel or any OPC point names, follow these steps:

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).



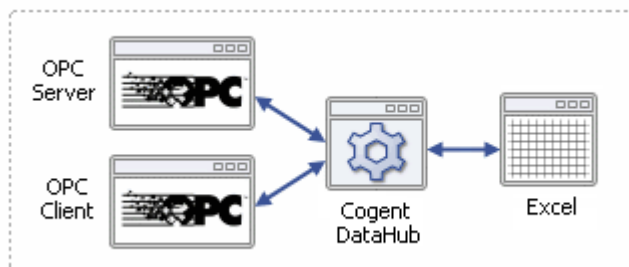
2. [Configure a point-to-point bridge](#) using the Excel point as the source and the OPC point as the destination. If we used our OPC server A and the point name switch from the example above, the bridge would look like this:



This bridge would allow you to turn the switch on and off (and thus control the motor) from an Excel spreadsheet.

Chapter 6. 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.



You can use [OPC Bridging](#) to link points from Excel to an OPC server or client without changing or assigning point names in Excel. Please refer to [Section 5.4.4, Bridging to Excel](#) for details.

6.1. 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 [Section 2.2, Test with simulated data](#).

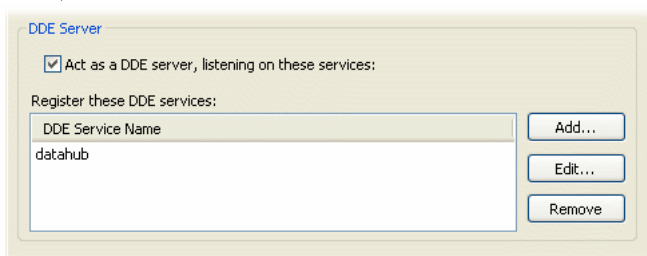
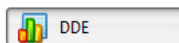
There are two ways to get data into Excel from the Cogent DataHub: by setting up a **DDEAdvise** loop to receive data [automatically](#), or by using a **DDERequest** command from a macro to [read](#) data. Deciding which to use depends on your situation. We suggest you become familiar with both. For more information about DDE and these commands, please refer to [Section 18.3.2, DDE Protocol](#) and [Appendix F, DDE Overview](#).

6.1.1. Method 1 - Drag and Drop using DDEAdvise

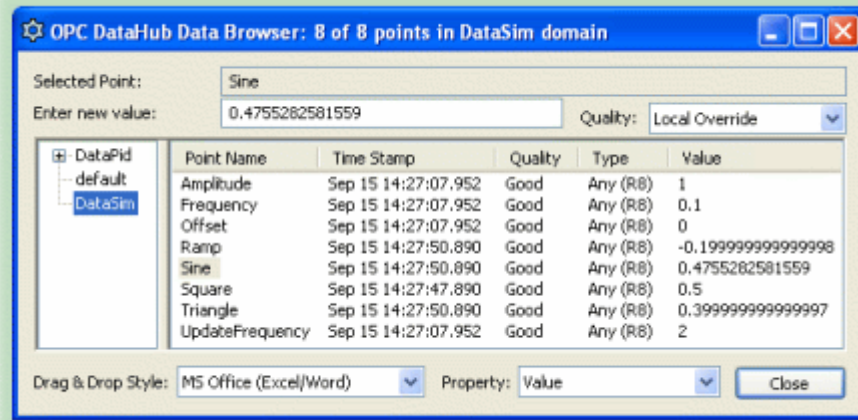
The easiest way to get data into Excel is to drag and drop point names from the DataHub Data Browser directly into the Excel spreadsheet. This automatically sets up a **DDEAdvise** loop between Excel and the DataHub. **DDEAdvise** loops update automatically so you will always see the latest data in your spreadsheet.

1. Right click on the Cogent DataHub system-tray icon and choose Properties.

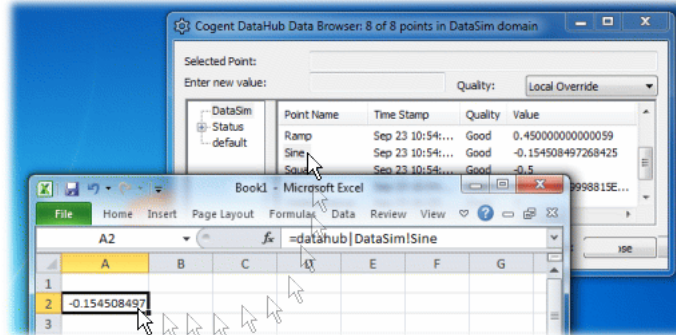
2. In the Properties window, select DDE



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 [Chapter 23, Troubleshooting](#) 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.

6.1.2. 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](#).

6.1.2.1. 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:

```
,
' 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*.

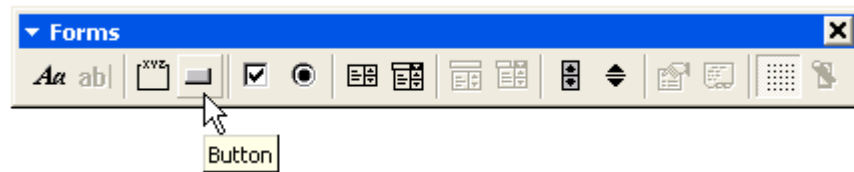


We use cell C2 in this example. If you need to use another cell, you will have to replace (2 , 3) with the row and column numbers of the cell you wish to use.

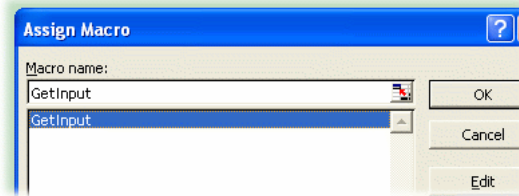
5. Save and close the Visual Basic text entry window.

6.1.2.2. Add a Control Button

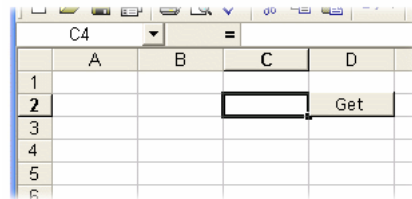
1. Activate the Forms toolbar by clicking on the View menu and selecting Toolbars, and then Forms.



2. Click on the button icon, and then click in cell D2. (We use this cell in our example, but you can choose another cell if you'd like.) An Assign Macro window should appear.
3. Select **GetInput** and click OK.



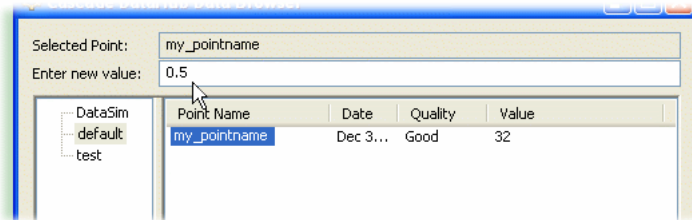
4. Change the label on the button to "Get".



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.

6.1.2.3. 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.

6.2. 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.

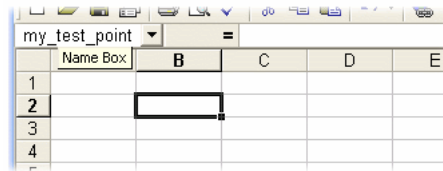
Each time data is sent for one point (data item), it is sent for all points. This can tie up your network if you have a large number of points. If you need to send data for a large number of cells, you can reduce this effect and reduce CPU load by sending a range that contains the cells.

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.

6.2.1. 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.

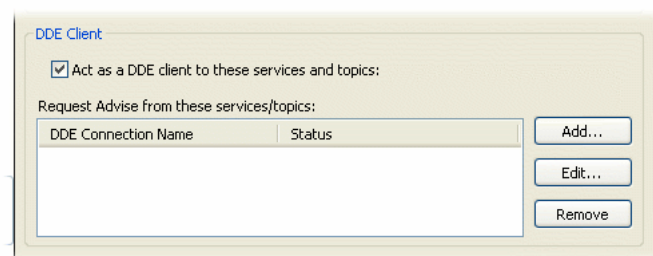
1. Open an Excel spreadsheet.
2. Choose a cell or range to hold the data you want to put into the DataHub. You will need to refer to your cell or range by row and column number, or by a name. For example, the cell B2 can be referred to as R2C2, or by giving it a name.



To name a cell or range, select it and enter a unique name in 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. 

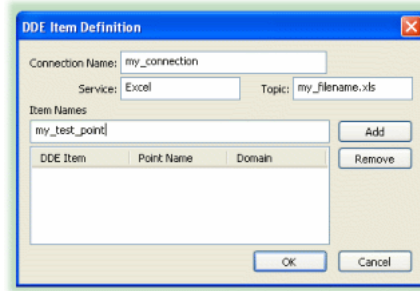


5. Make sure the Act as DDE client box is checked.



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:

Sheet in workbook	Service	Topic to enter
The first sheet	Excel	Test.xls
An unnamed sheet (e.g. Sheet2)	Excel	[Test.xls]Sheet2
A named sheet (e.g. StockData)	Excel	[Test.xls]StockData

- **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.

6.2.2. 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.

6.2.2.1. 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.

6.2.2.2. Add a Control Button



This explanation is illustrated in [Section 6.1.2.2, 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.

6.2.2.3. 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.

6.2.2.4. 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.

☐ Accept non-English characters in Excel strings (slower)

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:

```

'
' Cascade Multiple Writeback macro
'
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.

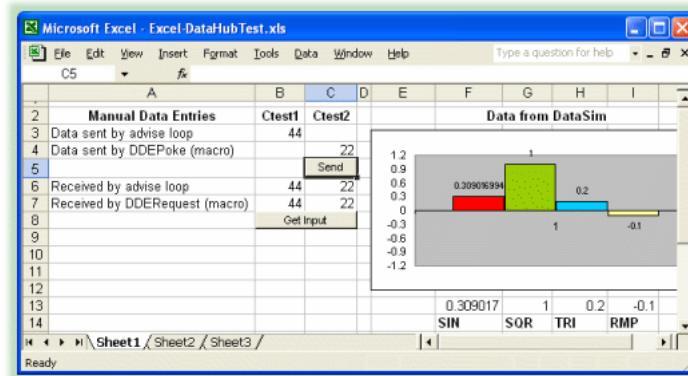
2. Create a separate 'open' and 'close' macro for the worksheet, and place the **DDEInitiate** and **DDETerminate** commands in those macros. This will keep communication to the DataHub open for the whole time the worksheet is open. The only drawback is that your data transmission could get interrupted (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.

6.3. Example

Here is an example of a worksheet that sends data points to the Cogent DataHub both ways: to the DataHub acting as a DDE client (using an advise loop), and by using macros (with **DDEPoke** and **DDERequest**). The worksheet also receives the same data back from the DataHub in both ways. For more information about DDE and these commands, please refer to [Appendix F, DDE Overview](#).



This worksheet, `Excel-DataHubTest.xls`, is included in your Cogent DataHub distribution.



Explanation of the pertinent manual data entry cells:

B3 The cell is named: Ctest1. When you use named cells to send data to the Cogent DataHub that is acting as a DDE client, each time you update any cell, all the other cells set up this way also send their data to the Cogent DataHub.

C4 Has a macro associated with it, shown below. When you use a macro to send data to the Cogent DataHub, each time you send updated data, only that cell's data gets sent to the Cogent DataHub.

The macro is:

```
Sub SendOutput()  
    mychannel = DDEInitiate("datahub", "default")  
    Application.Worksheets("Sheet1").Activate  
    Call DDEPoke(mychannel, "Ctest2", Cells(4, 3))  
    DDETerminate mychannel  
End Sub
```

C5 The Send button activates the macro associated with cell C4. The formula is:

Excel-DataHubTest.xls!SendOutput

B6 The cell formula is: =datahub|default!Ctest1

C6 The cell formula is: =datahub|default!Ctest2

B7 and C7 Have a macro associated with them, shown below. When you use a macro to receive data from the Cogent DataHub, the data is only updated once.

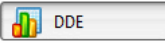
The macro is:

```
Sub GetInput()  
    mychannel = DDEInitiate("datahub", "default")  
    Application.Worksheets("Sheet1").Activate  
    newval1 = DDERequest(mychannel, "Ctest1")  
    newval2 = DDERequest(mychannel, "Ctest2")  
    Sheet1.Cells(7, 2) = newval1  
    Sheet1.Cells(7, 3) = newval2  
    DDETerminate mychannel  
End Sub
```

B8 and C8 The Get button activates the macro associated with cells B7 and C7. The formula is:

Excel-DataHubTest.xls!GetInput.

The Data from DataSim part of the spreadsheet receives the data from the four DataSim points (SIN, SQR, TRI, and RMP), and displays their values in a chart.

The Cogent DataHub DDE properties  for this example are set as follows:

- Act as DDE client to these services and topics: Checked.
- Connection Name: ExcelTest

- Service: Excel
 - Topic: Excel-DataHubTest.xls
 - DDE Item: Ctest1
 - Point Name: Ctest1
 - Data Domain: default
- Act as DDE server listening on these services: Checked.
 - DDE Service Name: datahub

6.4. 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 to maintain identical data sets.

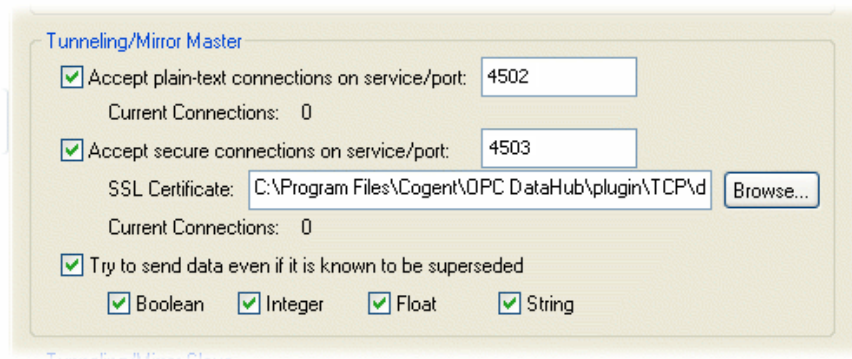


Mirroring is the same as *tunnelling*, as described in [Chapter 3, OPC Tunnelling](#). Mirroring can be also used to connect to Cascade DataHub running in Linux. Please refer to the Mirroring Data section of the Cascade DataHub for Linux and QNX manual for details.

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 [Section 20.4, 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.

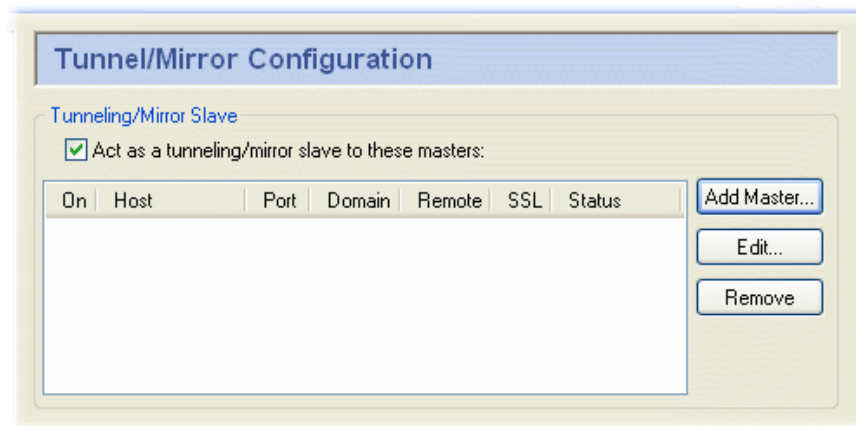
4. Click OK to close the Properties window.

You are now ready to configure the slave DataHub.

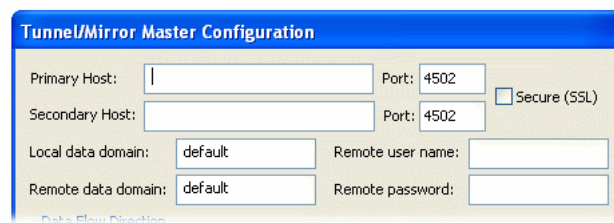
Configure the Cogent DataHub as tunnel/mirror slave

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  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 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.

- **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.



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

- **Primary Host:** `developers.cogentrts.com`
- **Port:** 4502
- **Local data domain:** `test`
- **Remote data domain:** `test`

6. You now have several options for the mirrored connection.

- 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 [Section 19.1, Binary Mode Tunnel/Mirror \(TCP\) Connections](#).

- **Target is a Cogent 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 for slow networks, please refer to [Section 19.2, Tunnel/Mirror \(TCP\) Connections for Slow Networks](#).

- **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.

6.5. 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.

6.5.1. 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

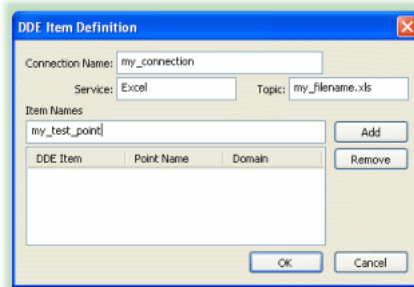
A DDEPoke command can be issued by Excel to send data to the Cogent DataHub based on a trigger within Excel. For this to work, the Cogent DataHub needs to be configured to [act as a DDE server](#) and have registered at least one service name. An Excel macro can then issue a DDEPoke to that service, 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:

`r2c5` - accesses the cell E2 (second row, fifth column)
`r3c3:r5c9` - accesses the range C3:I5
`MyRange` - accesses the cell or range that is named MyRange

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.

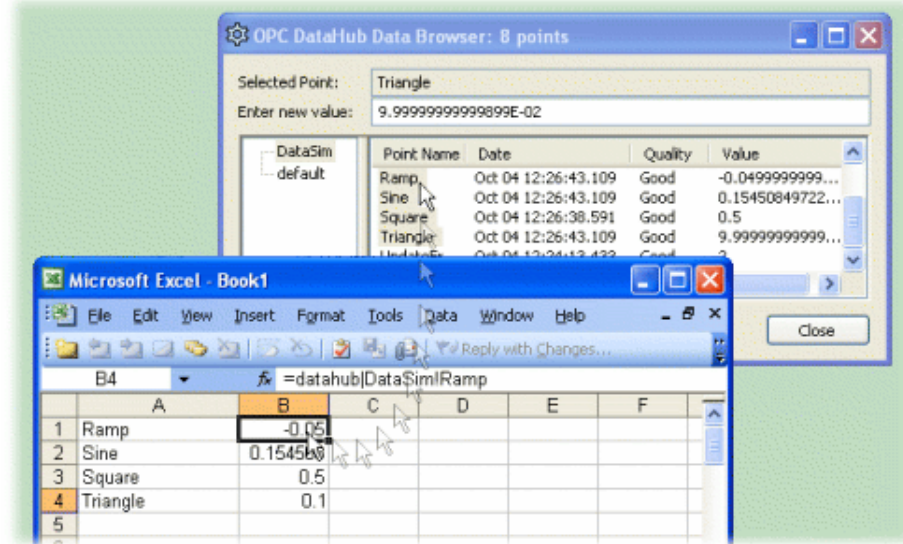
6.5.2. Getting a Range into Excel

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**.

	A	B	C
1	Ramp	-0.15	
2	Sine	0.404508	
3	Square	0.5	
4	Triangle	0.3	
5			

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:

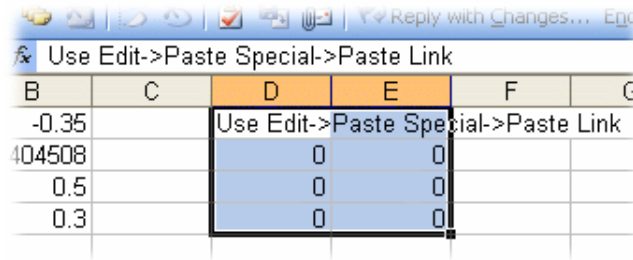
Point Name	Date	Quality	Value
FirstRange	Oct ...	Good	Ramp□0.35□□Sine□-0.404508497□□Square□0.5□□Tri...

The single boxes separate values in a row, and the double boxes separate rows. The array is now ready to put into Excel.

Drag and drop the array

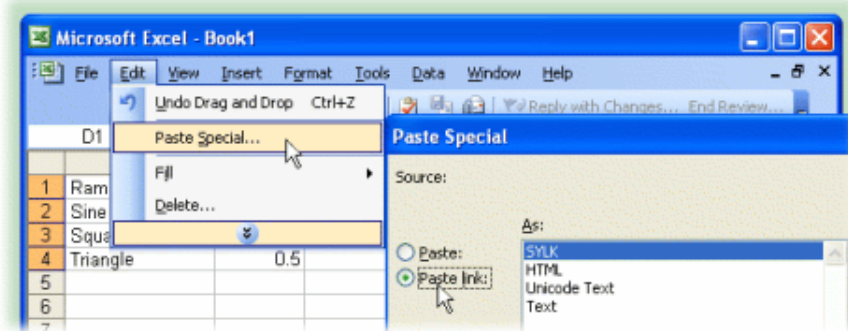
For simplicity's sake we are going to just put the same array back into Excel.

1. Click on the FirstRange point name, and drag it into Excel, dropping it in cell D1.



The values don't start updating right away because you have to tell Excel how to paste in the link.

2. 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.

6.5.3. 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.

```
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.

    DataArray = DDERequest(chan, DataPoint) ' request DataPoint from Channel

    ' 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 of 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

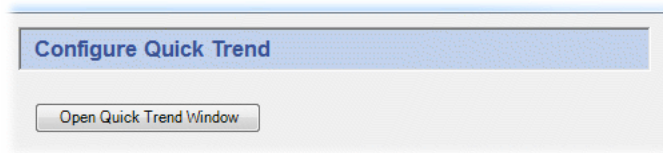
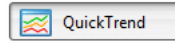
Sub GetData()
'
' This is a test function assigned to a button. It reads a test point into
' an arbitrarily sized matrix starting at A10
'
    Dim chan As Integer
    chan = DDEInitiate("datahub", "default")
    GetDataArray chan, "Sheet1", "TestArray", 10, 1
    DDETerminate (chan)
End Sub

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)
End Sub

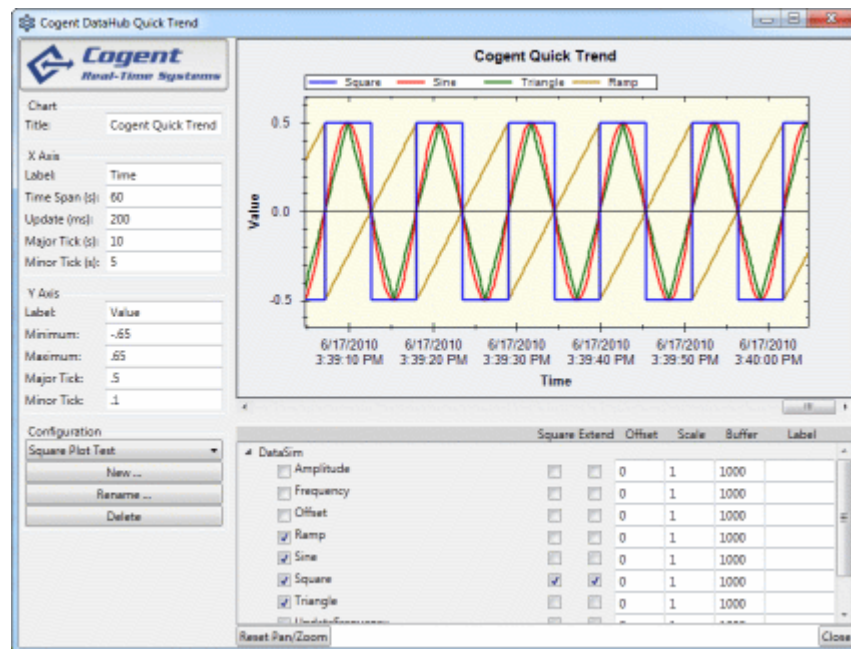
```

Chapter 7. Using QuickTrend

In the DataHub Properties window, select QuickTrend

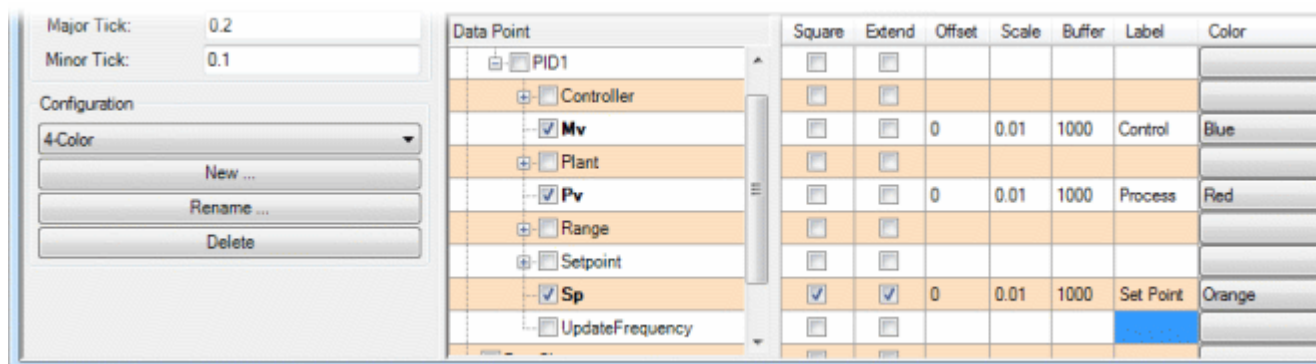


To configure a quick trend, press the Open quick trend window button:



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.

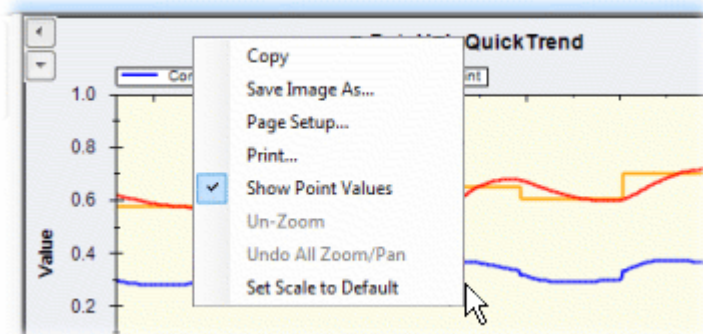
Label

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.
- For other features, the QuickTrend display has a menu available on a right mouse-click.

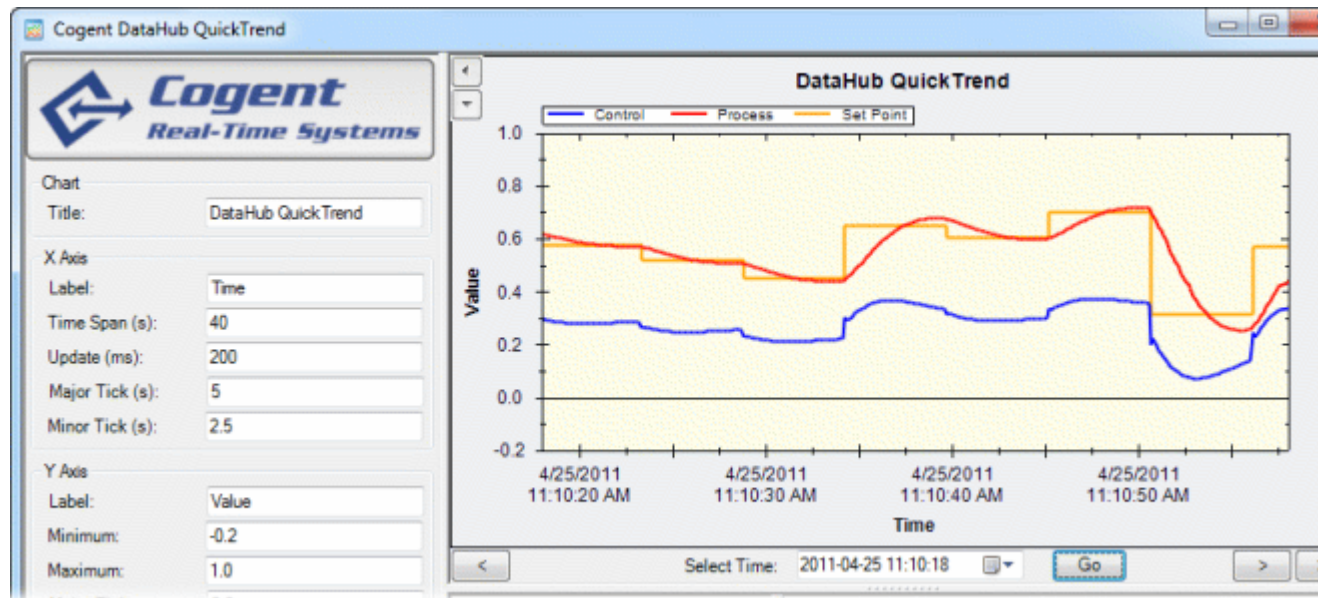


Using this menu you can copy, save, or print the current display, as well as unzoom and undo a zoom or pan. You can also hide or show point values, and reset the scale to the default.

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.

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.

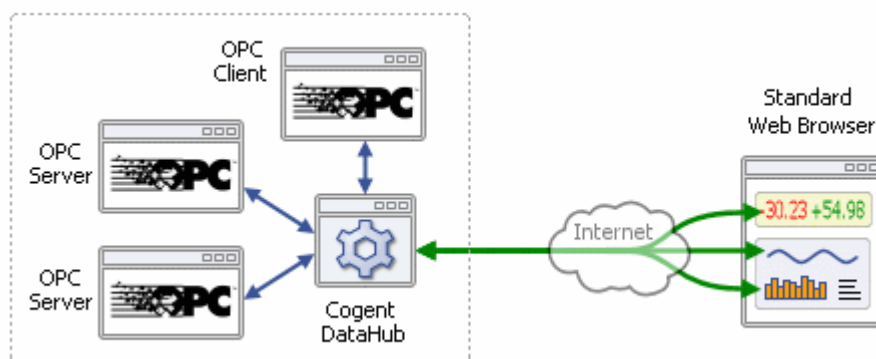
Minor Tick:

The value between minor tick marks.

Chapter 8. Using the Web Server

8.1. 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".
 - **Polling AJAX:** The Javascript code periodically sends commands to the DataHub to update the data on the browser page.
 - **Streaming AJAX:** The Javascript code maintains an open connection to the DataHub, which transmits updates to each point individually whenever its value changes.
- **Java applets** from the DataHub API for Java are embedded in the HTML code make a separate TCP connection from the browser to the DataHub. Once connected, the applets receive and display updated data values as soon as they change, without the need to refresh the page. The Java applets are executed by the browser, not by the DataHub.

Here is a comparison chart for these technologies:

	ASP	Polling AJAX	Streaming AJAX	Java applets
Web browser support	Desktop and mobile	Desktop and mobile	Desktop and mobile	Desktop only
Plug-in/Active X required	No	No	No	Yes, Java plug-in
Update speeds	No updates - Manual refresh required	Fast updates	Very fast updates	Very fast updates

	ASP	Polling AJAX	Streaming AJAX	Java applets
System requirements (CPU and memory)	Very low	High compared to Streaming AJAX and Java	Low compared to Polling AJAX, similar to Java.	Low compared to Polling AJAX, similar to Streaming AJAX.
Bandwidth requirements (Will depend on point count and update rate)	Very low	Relatively high	Moderate to low	Moderate to low
Security (Password / SSL protection)	Yes	Yes	Yes	Yes
Firewall friendly	Yes	Yes	Yes	No, requires firewall configuration
Licensing (Licenses required in addition to the standard DataHub Node license)	DataHub Web Server license	DataHub Web Server license	DataHub Web Server license, + TCP Link license for each connection	DataHub Web Server license, + TCP Link license for each connection
Programming Requirements	Uses DataHub scripting language	Uses JavaScript	Uses JavaScript	Uses HTML. Requires knowledge of Java to build custom applets.
Types of Application Common uses	Good for displaying static or slow moving data. Used for shift reports and statistics.	Good for displaying fast moving data and alarm conditions. Used in web monitoring and trouble shooting applications.	Excellent for displaying fast moving data. Used in remote monitoring and diagnostics systems.	Best choice for very high speed and large number of users. Used in stock trading and process control systems for monitoring and HMI displays.

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

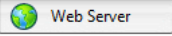
- AJAX automatically updates the web page whenever the data changes in the DataHub, no page refresh is required.
- AJAX can handle high speed updates to a large number of users. However, system resources on the server will increase as the number of points, the number of users and the speed of the updates 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, but some mobile web browsers may not support one of both types of AJAX display. We have found the Opera Mobile web browser to be one of the best for displaying AJAX on mobile devices.
- Polling AJAX and Streaming AJAX differ as follows:
 - Polling AJAX web pages make requests for new data to the DataHub on a polling cycle. This means there is usually a higher load on the CPU compared with Streaming AJAX, but in many cases the update speeds are comparable.
 - Streaming AJAX web pages are highly efficient at moving data, which means they use very little system resources and much less network bandwidth compared to Polling AJAX applications.

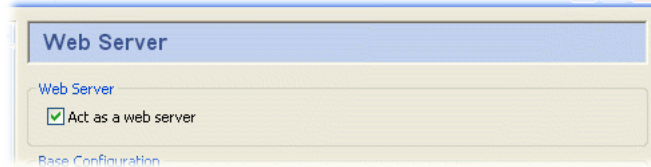
More about Java applets

- Java applets are the best choice for displaying high speed data in complex web graphics like gauges, trends and progress bars.
- Java displays can typically handle a large number data points, updating rapidly to a large number of users.
- In most cases, Java web pages use far less system and network resources than AJAX pages.
- Java pages are often used for displaying executive summary screens for management, or as an HMI to remote facilities.
- We provide a wide range of Java applets that allow you to create sophisticated web pages with no programming required. Just use a simple HTML editor to configure the applet code and you can have a working web page up and running in a few minutes.
- Java pages can be displayed using most modern desktop browsers, but they do require the free Java plug-in to be installed first. The plug-in will download and install automatically if it is not already installed on your system.
- We have found that Java is generally not supported on mobile web browsers.

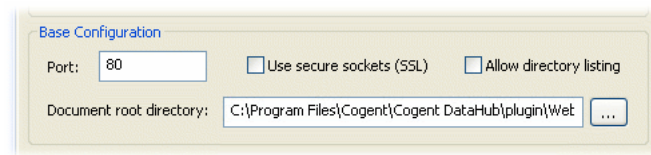
8.2. Configuring the Web Server

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:

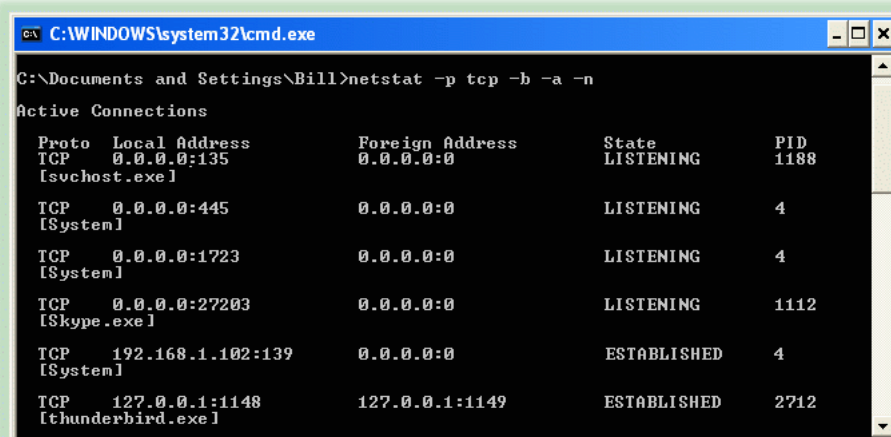


Windows allows multiple users on a single TCP port, and never refuses a connection. However, this can cause irregular behavior. It is essential that the DataHub Web Server be the exclusive user of a port.

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:

```
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

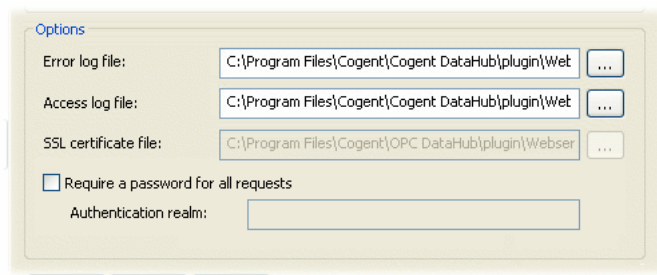
State column shows the state of that process. The only state we are interested in is LISTENING. Whatever port you are using for the DataHub Web Server, it should be the only process on that port.

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).

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.

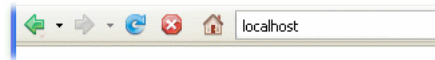
Authentication realm

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

8.3. Viewing the Web Demos

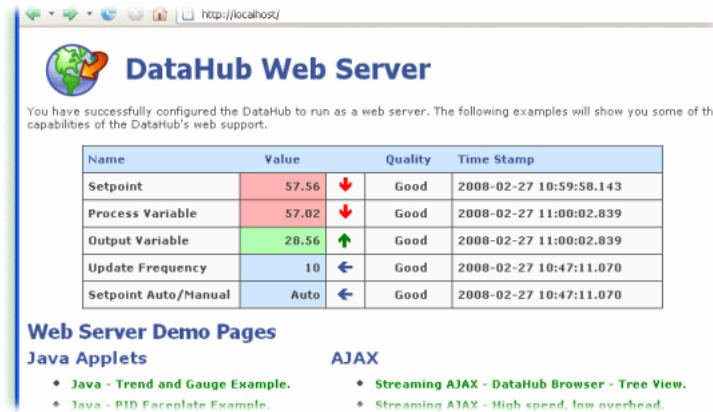
This is how to access the demo page that comes with your DataHub installation:

1. Ensure that the DataHub is running, and that the DataHub Web Server is enabled.
2. Type the IP address of the DataHub into the Address field at the top of your web browser.



If you are running the DataHub and web browser on the same machine, type `localhost`. Otherwise, type the IP address or computer name of the computer running the DataHub.

3. The welcome page should appear. You can follow the links to see the various demos. The live data in the demos comes from DataPid. If you don't see data updating, you'll need to start DataPid.



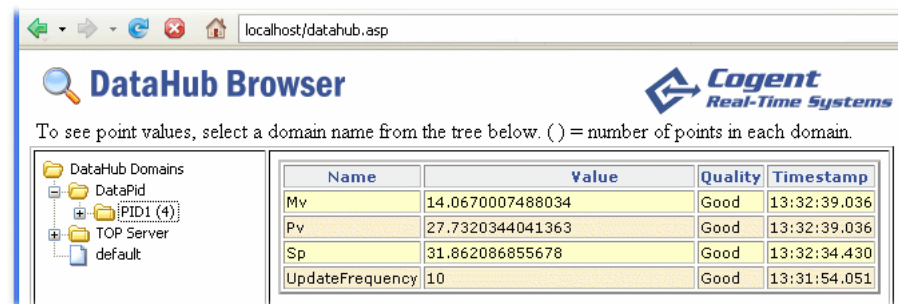
8.4. Viewing Your Own Data

8.4.1. DataHub Browser

You can view all the data points currently in the DataHub using the DataHub Browser. Here's how to view the data on your system:

1. Ensure that the DataHub is running and connected to your data sources, and that the DataHub Web Server is enabled.
2. If you are running your web browser on the same machine as your DataHub, type `localhost/datahub.asp` into the Address field at the top of your browser. If you are running your web browser on a different machine from the DataHub, type the IP address or computer name of the computer running the DataHub, instead of `localhost`.

The Web Data Browser page should display:



The DataHub domains and data hierarchy are displayed in the left pane, and live updates of the point values for the selected part of the hierarchy are shown the right pane. You can modify the header for

this page by editing this file: C:\Program Files\Cogent\Cogent DataHub\Plugin\Webserver\html\domaintreeheader.asp.

8.4.2. DataHub Table View

You can also view all the data points in a single data domain on your system, using the DataHub Table View page, as follows:

1. Ensure that the DataHub is running and connected to your data sources, and that the DataHub Web Server is enabled.
2. Open a web browser on this computer or another client computer and type the following URL into the address bar:

```
http://computer address/table.asp?parent=domain name[:branch][.sub-branch]
```

where

computer address

The IP address or network name of the computer running the Cogent DataHub.

domain name

The name of the data domain that contains the data you want to display.

branch

The branch (if any) containing the data you want to display.

sub-branch

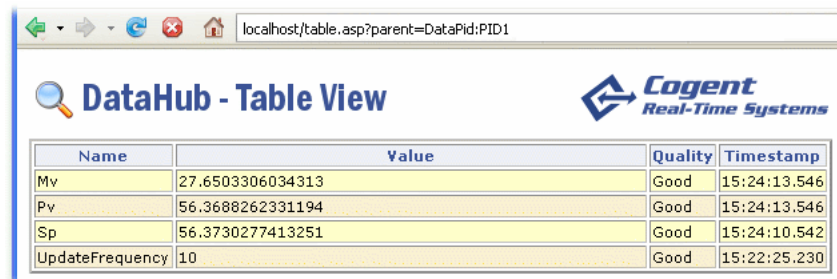
The sub-branch (if any) containing the data you want to display.

Sometimes the organization of the data within the Cogent DataHub can be quite complex, including branches and sub-branches. This is often the case when reading data from a data source that has a multi-level data hierarchy. In these cases we reference locations within the data structure using the branch and sub-branch syntax shown above.

3. Examples:

- If you are currently running the DataPid program, this URL:

`http://localhost/table.asp?parent=DataPid:PID1` will display this:



Name	Value	Quality	Timestamp
Mv	27.6503306034313	Good	15:24:13.546
Pv	56.3688262331194	Good	15:24:13.546
Sp	56.3730277413251	Good	15:24:10.542
UpdateFrequency	10	Good	15:22:25.230

- An entry like this:

```
http://192.168.5.55/table.asp?parent=Plant_1
```

Would display data from the Plant_1 data domain of the DataHub.

- An entry like this:

```
http://PlantServer/table.asp?parent=Section_1:Zone_1
```


Would display data from the Zone_1 branch within the Section_1 data domain of the DataHub.

- An entry like this:

```
http://PlantServer/table.asp?parent=Section_1:Zone_2.Temp.Setpoints
```

Would display data from the Setpoints branch, three levels down the tree structure within the Section_1 data domain of the DataHub.

- For any of these entries, you can just type **localhost/...** if you are running the web browser on the DataHub computer, like this:

```
http://localhost/table.asp?parent=Area2
```

4. You can change the header image for this page by editing this file: C:\Program Files\Cogent\Cogent DataHub\Plugin\Webserver\html\table.asp. The line to change is clearly marked:

```
<!-- ***** Logo Image Goes Here ***** -->

<br />

<!-- ***** -->
```

5. You can also add page content below the table by putting HTML code after the final `</script>` tag, towards the bottom of the page:

```
...
</script>
```

(insert extra HTML here)

```
</body>
</html>
```

8.5. Modifying the ASP and AJAX Demo Files



To modify the Java demos, you need to use the DataHub API for Java.

You can use the demo files that come in the Web Server archive as a basis for creating your own web pages. The default location is here:

```
C:\Program Files\Cogent\OPC DataHub\Plugin\Webserver\html\
```

Both ASP and AJAX use .asp files to build web pages. The .asp files in the DataHub distribution contain embedded DataHub scripts which write parts of the final .asp page that gets rendered. In other words, what you see when you select **View Source** in your browser is not the original .asp file, but the results of processing it. For example, this line of code in the DH_asp_2.asp file:

```
<td align="center"><@ quality @></td>
```

writes this on the displayed DH_asp_2.asp page:

```
<td align="center">Good</td>
```

Each .asp file can contain two types of special markup syntax for inserting DataHub script code, each with its own purpose.

- `<@ ... @>` indicates an expression that gets evaluated when the page is created. The results of the expression get written into the page. An example of this would be inserting the value of a DataHub point.
- `<% ... %>` indicates an expression or statement that is also evaluated when the page is created, *but* nothing gets written to the page. An example would be a function that can be used by other code in the page.

Example

The `DH_asp_2.asp` file is shown below. It contains two functions: `TimeStamp`, for calculating time stamps; and `NewSetPoint` to send a new `Setpoint` value back to the DataHub. It also contains a `for` loop for creating the table rows. All of these are marked up with the `<% ... %>` syntax, while the values that get displayed in the table are marked up with the `<@ ... @>` syntax.



The script code also uses a `try / catch` set of statements for error handling. For more information, please refer to `try catch` in the Gamma documentation.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<head>
  <title>DataHub Web Reports</title>
  <link rel="stylesheet" type="text/css" href="css/dhwebserver.css">
</head>
<html>
<body>

<%
// Load files we need
require ("Quality");
require ("Time");

// DataHub script function to generate a text timestamp with milliseconds
local TimeStamp;
function TimeStamp (wintime)
{
  local    unixtime = WindowsTimeToUnixTime (wintime);
  local    tm = localtime (unixtime);
  format ("%d-%02d-%02d %02d:%02d:%03d",
    tm.year + 1900, tm.mon+1, tm.mday,
    tm.hour, tm.min, tm.sec, (unixtime % 1) * 1000);
}
// DataHub script function to send new Setpoint value back to the DataHub.
// This is done by refreshing the page with a new URL that appends the new Setpoint
// value to the end, for example, http://localhost/DH_asp_2.dhht?Setpoint=12
// This is interpreted by the DataHub web server and passed to an internal function
// that writes the value to the DataHub. The web server then returns the refreshed page
// contents to the browser, which includes the new value for the setpoint.
//
function NewSetPoint()
{
  local    args, i, sp;

  args = list_to_array(string_split(QUERY_STRING, "&?", 0));
  for (i=0; i<length(args); i++)
  {
    if (args[i] == "Setpoint" && (sp = number(args[i+1])) != 0)
    {
      $DataPid:PID1.Sp = sp;
    }
  }
  <meta http-equiv="refresh" content="0;URL=<%= REQUEST_URI %>"/>
}
}

NewSetPoint();
```

```

%>
<span style="font-size:12px; font-weight:bold; color:#395294">This is a Static data display.</span>
<p></p>

<table border="1">
<tr><th width="70">Name</th><th width="60">Value</th><th width="60">Quality</th><th width="90">Timestamp</th>

<!-- The following ASP code uses a loop to generates the table of DataHub point values -->
<%
    // List of DataHub points to be displayed
    local points = [#$DataPid:PID1.Sp, #$DataPid:PID1.Pv, #$DataPid:PID1.Mv ];

    // List of display names for each point in the list
    local dispname = [ "Setpoint", "Process", "Output" ];

    // List of units to be displayed for each point value
    local units = [ "", "", "" ];
    local value, item, quality, timestamp, i;
    for (i=0; i<length(points); i++)
    {
        try
        {
            item = getprop (points[i], #__datahub_point__);
            if (item)
            {
                value = string (eval (points[i]), " ", units[i]);
                quality = GetQualityName(item.quality);
                timestamp = TimeStamp(item.timestamp);
            }
            else
            {
                value = "Please run DataPid";
            }
        }
    }
%>
    <tr>
        <td align="left" style="color:#395294; font-weight:bold"><@ dispname[i] @></td>
        <td align="center" style="color:#008800; font-weight:bold"><@ format("%.2f",number(value)) @></td>
        <td align="center"><@ quality @></td>
        <td align="center"><@ substr(timestamp,11,-1) @></td>
    </tr>
<%
    }
    catch
    {
        %>
        <@ _last_error_ @>
        <%
    }
}
%>
</table>
<p></p>

<table width="320" border="0" cellpadding="5" cellspacing="5">
<tr>
<td valign="middle"><strong>Enter new Setpoint:</strong></td>
<td width="130" valign="middle">
<form style="margin:0px;">
<input name="Setpoint" type="text" id="Setpoint" size="4">
<input type="submit" id="submit" value="Submit">
</form>
</td>
</tr>
<tr>
<td colspan="2">
<div class="strongBlue">Refresh this page to see new values.</div>
New Setpoint values are being auto-generated by the DataPid program.

```

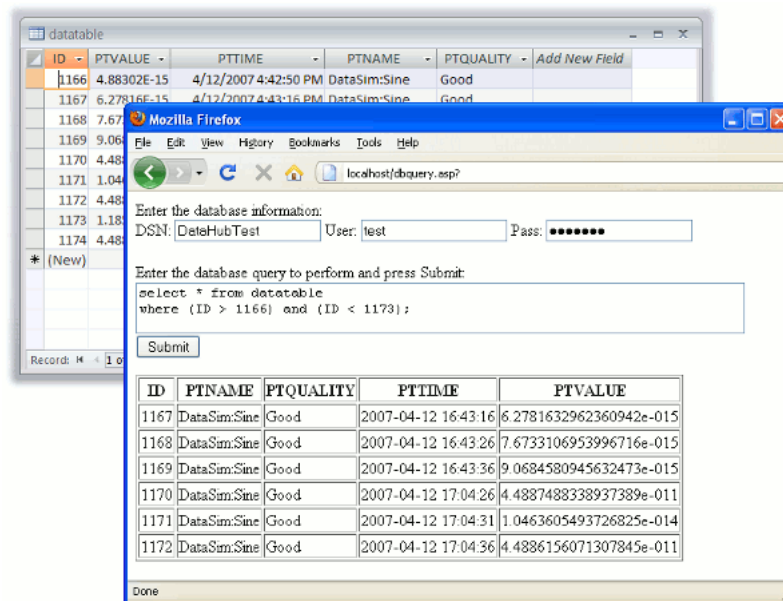
```

        </td>
    </tr>
</table>
<p></p>
<a href="index.asp"><-- Go back</a>
</body>
</html>

```

8.6. 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\Cogent\OPC 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 ()
{
    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 %>">
<br>
<br>
Enter the database query to perform and press Submit:
<br>
<textarea name="Query" wrap="logical" rows="10" cols="80">
<%= Query %></textarea>
<br>
<input type="submit" id="Submit" value="Submit">
</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);
%>

</body>
</html>

```

8.7. 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.

8.7.1. Streaming XML How-To

8.7.1.1. Built-in Streaming Data

The built-in streaming data is accessed using a URL of the form:

`http://hostname: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
```

`head=file name`

The head is the name of a file relative to the document root of the DataHub Web Server that will be transmitted once when the connection is first made. This can contain information like XML version or web page header information. Example:

```
head=my_header_file
```

`tail=file 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 connection 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 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>
```

`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 connection. 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&msglimit=30&throttle=0.2&template=xml`

The resulting output would look like this:

```
<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="1275683573.1379995" />
<point name="DataPid:PID1.Pv" value="57.5623437038486" type="1" quality="192"
timestamp="1275683573.1379995" />
</points>
<points>
<point name="DataPid:PID1.Mv" value="39.2907100165968" type="1" quality="192"
timestamp="1275683573.3569999" />
<point name="DataPid:PID1.Pv" value="59.5695627360927" type="1" quality="192"
timestamp="1275683573.3569999" />
</points>
<points>
<point name="DataPid:PID1.Mv" value="40.0353526249595" type="1" quality="192"
timestamp="1275683573.5760002" />
<point name="DataPid:PID1.Pv" value="61.5352593843648" type="1" quality="192"
timestamp="1275683573.5760002" />
</points>
<points>
<point name="DataPid:PID1.Mv" value="40.6012237044704" type="1" quality="192"
timestamp="1275683573.7950006" />
<point name="DataPid:PID1.Pv" value="63.4279691691699" type="1" quality="192"
timestamp="1275683573.7950006" />
</points>

```

8.7.1.2. 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:

```

<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:

```

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:

```
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>\n",
                                             "</data>\n",
                                             0, 0.0,
                                             "<?xml version='1.0' encoding='UTF-8'?'>\n",
                                             "",
                                             "<point name=%s value=%w type='%d' quality=%d times-
tamp='%g' />\n",
                                             "<point name=%s value=%s type='%d' quality='%d' times-
tamp='%g' />\n",
                                             //"<point name=%s value=%s type='%d' quality='%d' times-
                                             " ");
```

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 OPC DataHub, this argument may be missing.
`separator` - as above

5. Ensure that your new script runs when the DataHub starts.

6. Run your script. You should now be able to query data from a URL like:

```
http://localhost/stream?name=DataPid&children=1&recursive=1&template=my_custom
```

8.7.2. 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.

8.7.2.1. 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://hostname: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_name&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>
```

8.7.2.2. 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 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 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>

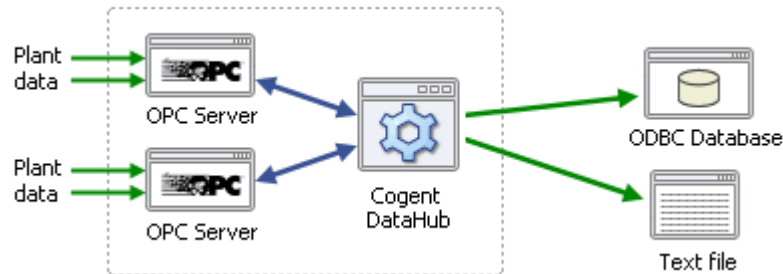
In this example, the delimiters `<% %>` are used to indicate Gamma script code that will not generate a replacement string. The Gamma script between `<%` and `%>` must be one or more Gamma statements, not expressions. Notice that Gamma statements may be broken up across occurrences of the `<% %>` delimiters, 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.

Chapter 9. Write to a Database

9.1. Introduction

The Cogent DataHub can write data to any ODBC compliant database or text file¹.



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 refer to Tutorial 3: Writing data from a database to the DataHub in the DataHub ODBC Scripting manual.

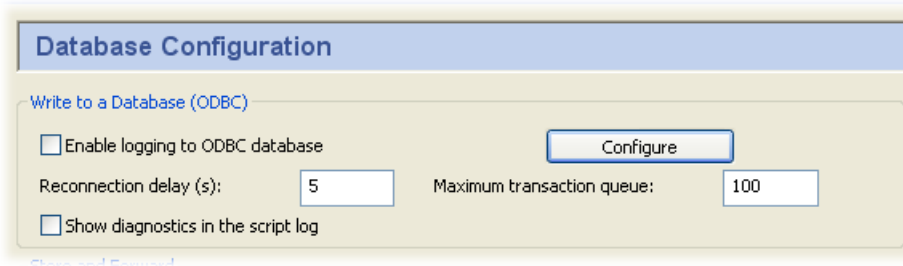
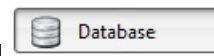
The Cogent DataHub's ODBC Data Logging interface provides an easy way to connect to a DSN, create or select a table, assign data point properties to table columns, and assign a trigger and conditions for logging points. The fastest way to learn how to use the interface is by watching the web-site video or by using the [Quick Start](#).

9.2. 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.

Open the ODBC Data Logging window

1. In the Cogent DataHub Properties window, select Data Logging
2. Click the Configure button.



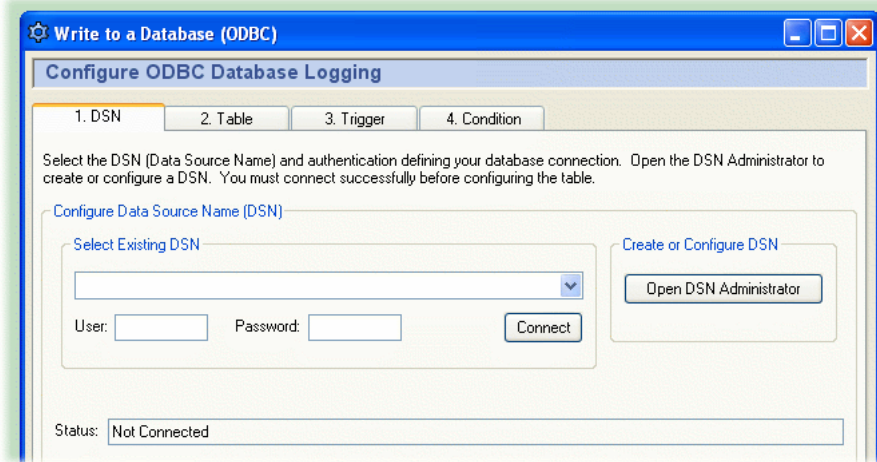
This opens the ODBC Data Logging window, shown below.



The Data Logging Configuration interface is explained in detail in [Section 9.3, Configuring the Queue, Store and Forward](#). For now, you can use the defaults.

Connect to the Database

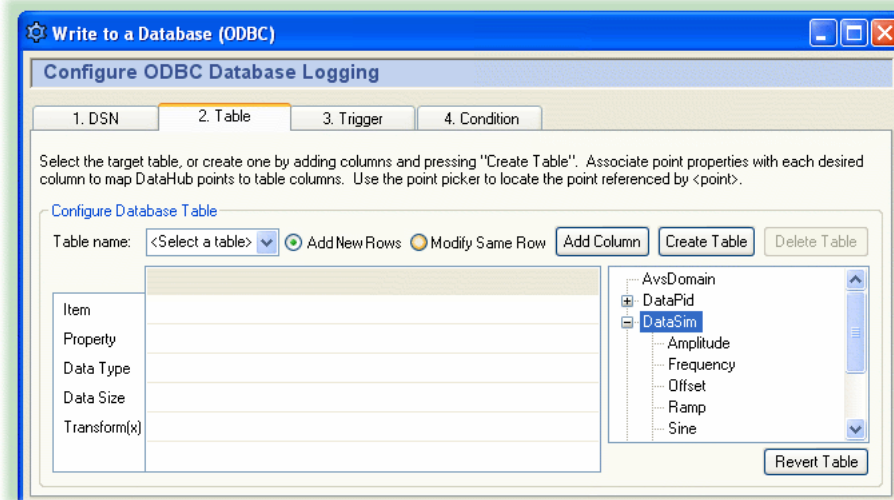
1. Select the 1. DSN tab. A DSN is a Data Source Name. Windows uses this name to identify the database you want to connect to.



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 appropriate 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.

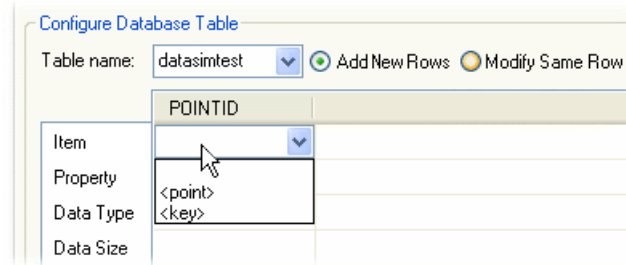
Configure a Table

1. Select the 2. Table tab.



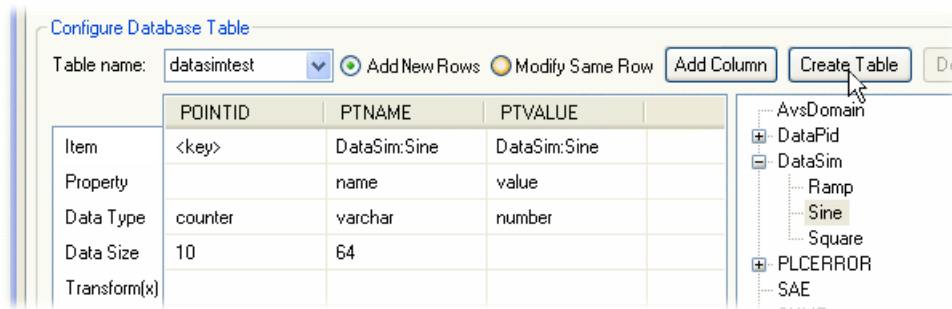
2. Start the DataSim program if it isn't already running, and ensure that it is connected to the DataHub.
3. In the Table name field type: `datasimtest`.
4. Click the Add Column button, type: `POINTID` in the pop-up dialog, and then click OK.

- Click under the POINTID label in the Item row.



- Select <key> from the drop-down list. Notice after you make your selection that the word counter gets entered automatically for the Data Type.
- Click the Add Column button and enter the name PTNAME.
- In the point-picker list on the right, expand the DataSim data domain and select the point named Sine.
- 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**.
- 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,

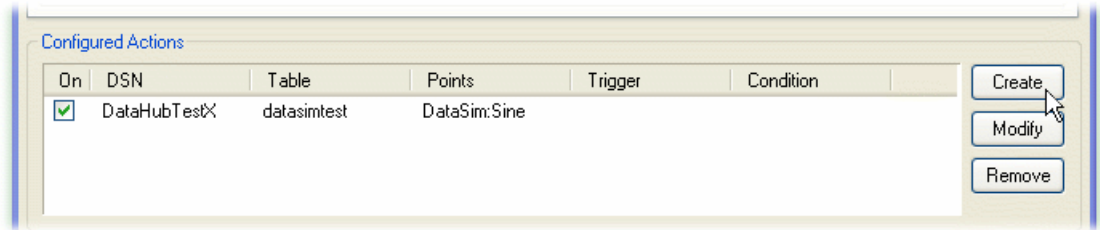
The entry fields should now look similar to this:



- 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 [Section 9.5, Configuring a Database Table](#).

- When you have the table the way you want it, go down to the **Configured Actions** box and click the **Create** button.



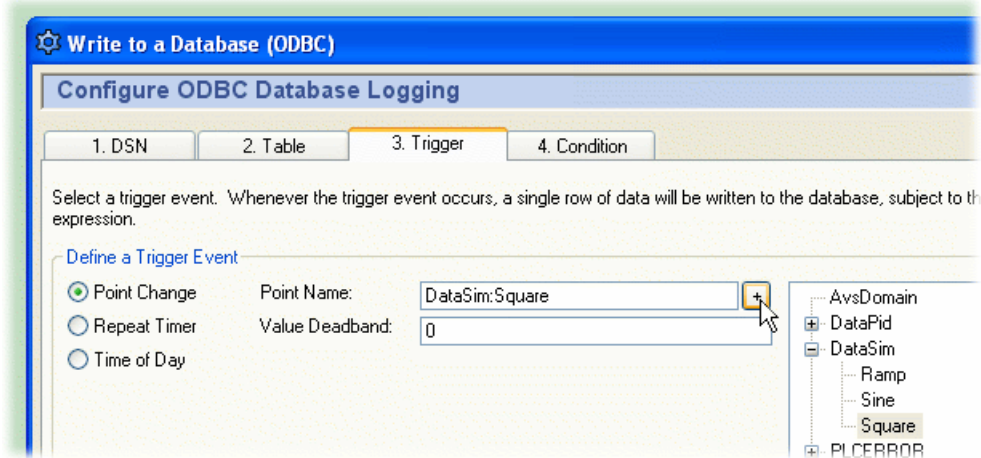
A new configured action should appear in the list. For more information about configured actions, please refer to [Section 9.9, Configured Actions](#)

Next, to get the DataHub to write the data, you need to assign a trigger.

Assign a Trigger

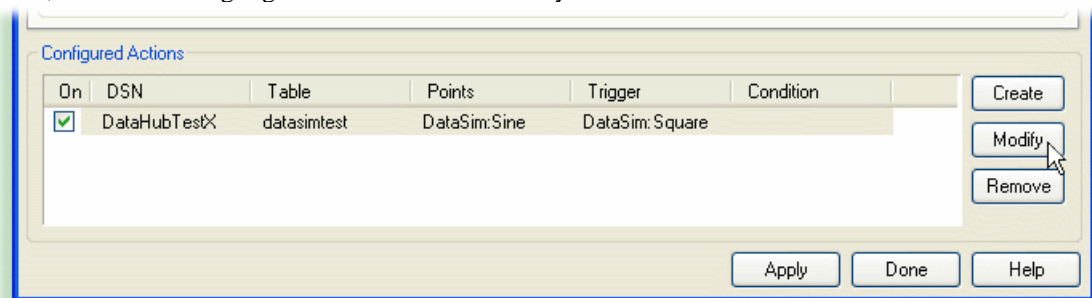
For this example, we will trigger the action whenever the DataSim:Square point changes value.

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.



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 [Section 9.7, 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.



Your configured action should now display the DataSim:Square point as your Trigger.

5. Click the Apply button to activate the configured action.

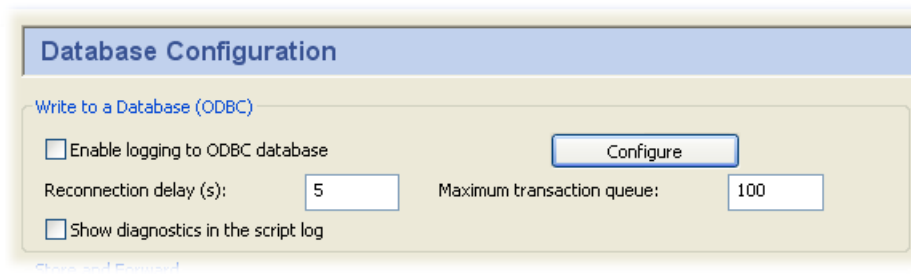
- Open the [Script Log](#) to check for error messages and ensure that your data is being written successfully. Each write action to the database gets logged here. You can also verify the writes by querying the database itself.

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.

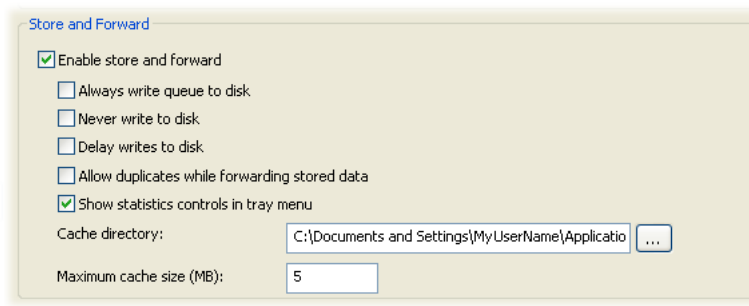
9.3. Configuring the Queue, Store and Forward

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.



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).

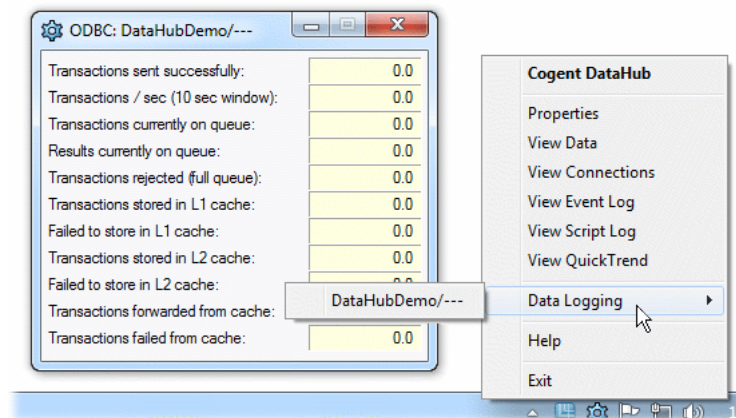
Allow duplicates while forwarding stored data

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.

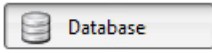
Maximum cache size (MB):

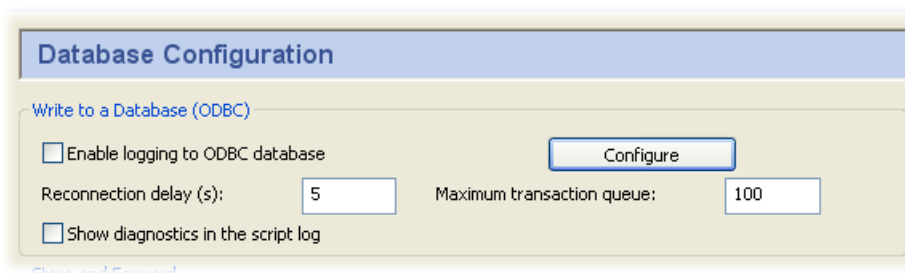
The amount of disk space to allocate for the cache, in megabytes.

9.4. 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

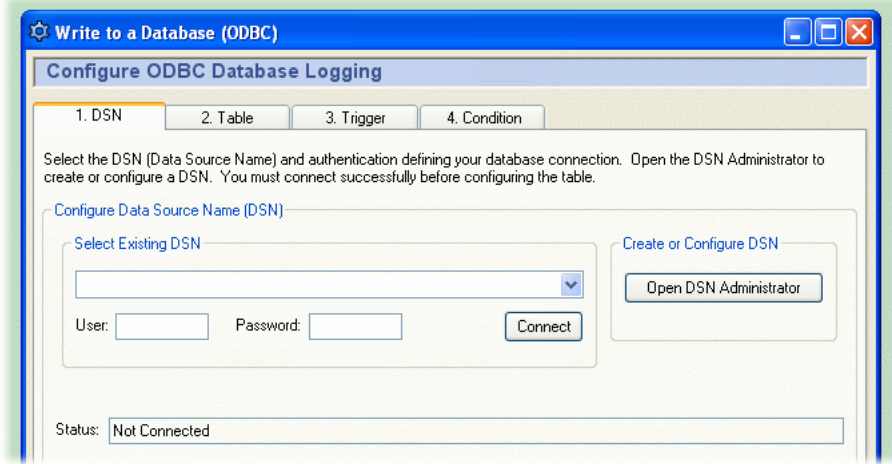
1. In the Cogent DataHub Properties window, select **Data Logging** .
2. In the Configure Database (ODBC) Data Logging section click the **Configure** button.





The Data Logging Configuration interface is explained in detail in [Section 9.3, 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 appropriate user name and password.



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.

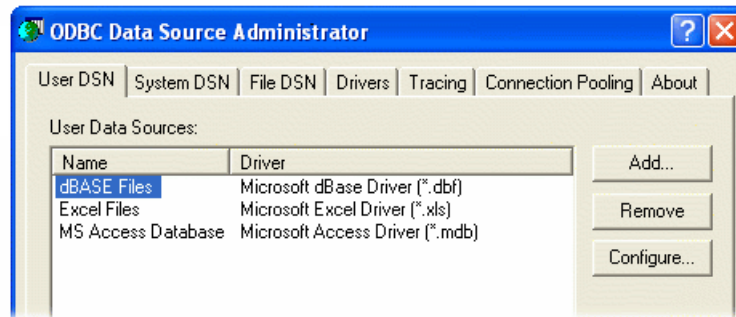
Creating or Configuring a DSN



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.

9.5. 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.

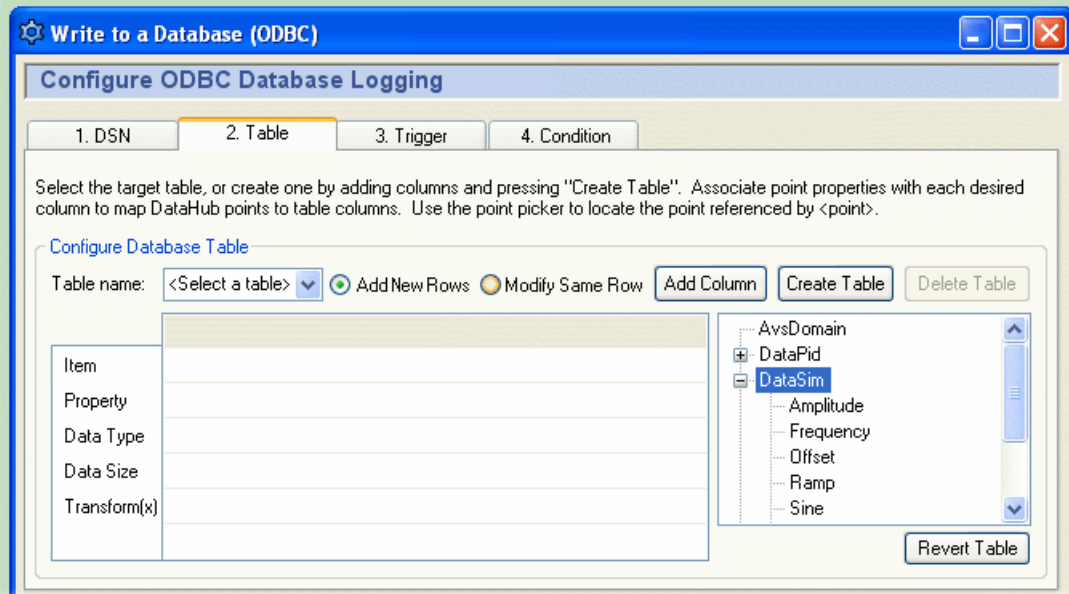


Table Selection or Creation

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 column 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-incrementing. Please refer to [Section 9.6, 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 [Section 9.6, 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>

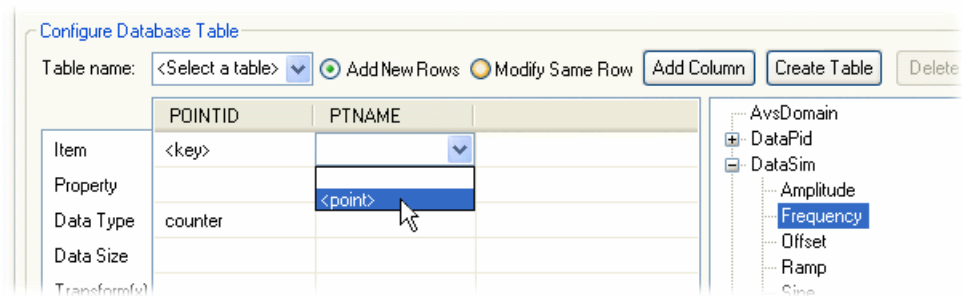
Data Type: The appropriate type for your database (typically VARCHAR, NVARCHAR or CHAR).



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:

	Column A
Enter for Property	timestamp

Enter for Data Type	datetime or timestamp or date
---------------------	-------------------------------

- Databases like MS Access:

	Column A	Column B
Enter for Property	timestamp	timems
Enter for Data Type	datetime	number or integer

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.

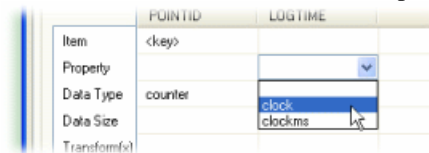
clockms

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:

System Time	Point1	Point2	Point3
08:12:56.000	43.883	3.727	213.905

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 * 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.

9.6. 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 existing 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

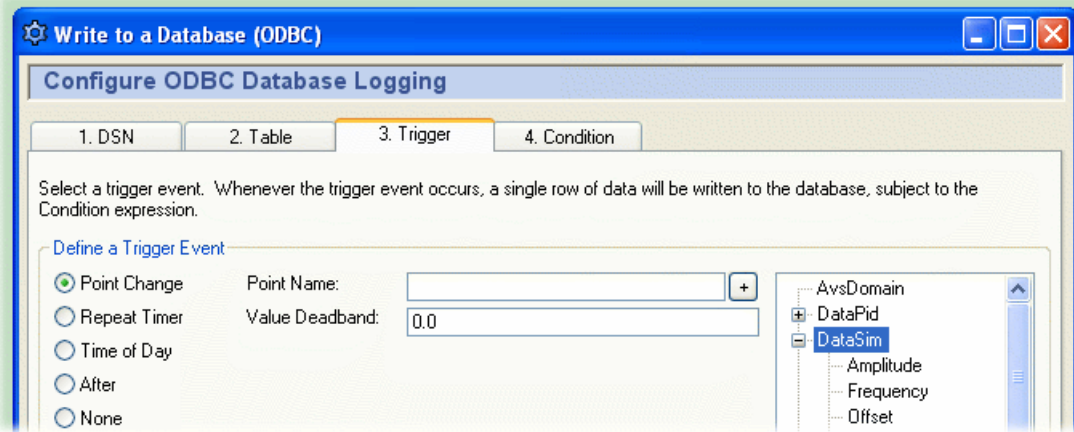
If you choose to modify a row in the database, such that any new data will overwrite the existing data in the row, you must be able to uniquely identify that row. This means that you **must** have a key column.

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.

9.7. 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 [Section 9.8, 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 [Section 9.8, Setting Trigger Conditions](#).

The ranges of the fields are:

Year:	1970-*	Hour:	0-23
Month:	1-12	Minute:	0-59
Day:	1-31	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:

Year:	<input type="text" value="*"/>	Hour:	<input type="text" value="8"/>
Month:	<input type="text" value="*"/>	Minute:	<input type="text" value="45"/>
Day:	<input type="text" value="*"/>	Second:	<input type="text" value="0"/>

would cause a row to be logged at 8:45 every day, every month, and every year.

- These entries:

Year:	<input type="text" value="*"/>	Hour:	<input type="text" value="*"/>
Month:	<input type="text" value="*"/>	Minute:	<input type="text" value="0"/>
Day:	<input type="text" value="15"/>	Second:	<input type="text" value="0"/>

would cause a row to be logged every hour on the 15th day of each month, every year.

- These entries:

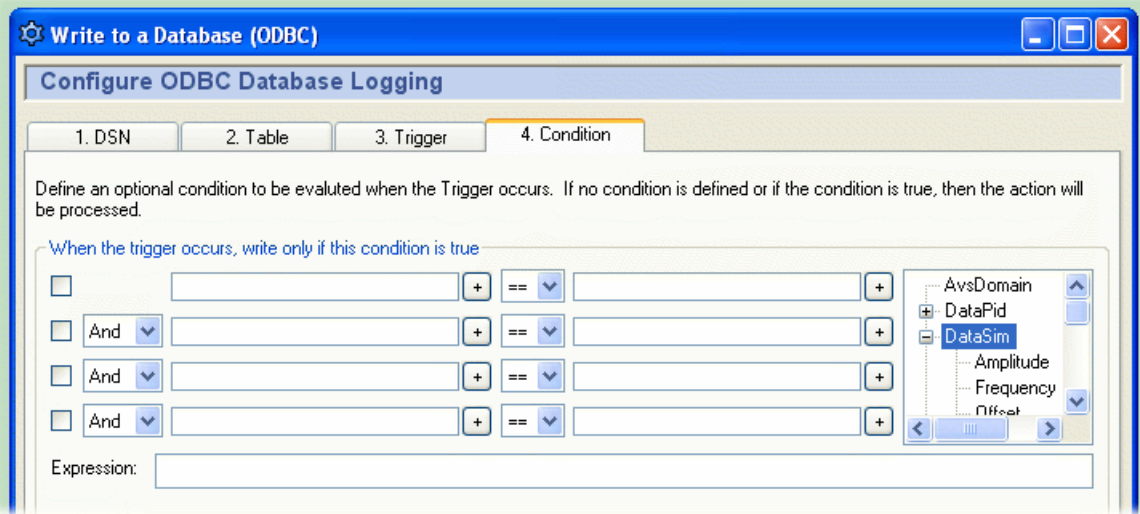
Year:	<input type="text" value="*"/>	Hour:	<input type="text" value="8,10,12,14,16,18"/>
Month:	<input type="text" value="*"/>	Minute:	<input type="text" value="0-4"/>
Day:	<input type="text" value="*"/>	Second:	<input type="text" value="0"/>

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.

9.8. 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:

When the trigger occurs, write only if this condition is true

<input checked="" type="checkbox"/>		\$DataSim:Sine	+	>	0
<input checked="" type="checkbox"/>	And	thistrigger	+	>	lasttrigger
<input type="checkbox"/>	And		+	==	
<input type="checkbox"/>	And		+	==	

Expression: `[$DataSim:Sine > 0 && thistrigger > lasttrigger]`

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:

When the trigger occurs, write only if this condition is true

<input checked="" type="checkbox"/>		<code>localtime(clock()).wday</code>	+	>	0
<input checked="" type="checkbox"/>	And	<code>localtime(clock()).wday</code>	+	<	6

would create this Gamma code::

```
(localtime(clock()).wday > 0 && localtime(clock()).wday < 6)
```

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:

<code>.sec</code>	The number of seconds after the minute (0 - 59).
<code>.min</code>	The number of minutes after the hour (0 - 59).
<code>.hour</code>	The number of hours past midnight (0 - 23).
<code>.mday</code>	The day of the month (1 - 31).
<code>.mon</code>	The number of months since January (0 - 11)
<code>.year</code>	The number of years since 1900.
<code>.wday</code>	The number of days since Sunday (0 - 6).
<code>.yday</code>	The number of days since January 1 (0 - 365)
<code>.isdst</code>	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 [Section 9.7, 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.

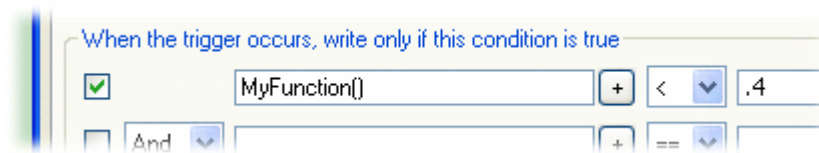
Custom Conditions

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 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 ()
{
    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 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 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;
    }
}

```

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 included 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:

When the trigger occurs, write only if this condition is true

<input checked="" type="checkbox"/>		HasValueChanged(##DataPid:PID1.:	+	!=	▼	nil
<input type="checkbox"/>	And ▼		+	==	▼	
<input type="checkbox"/>	And ▼		+	==	▼	
<input type="checkbox"/>	And ▼		+	==	▼	

Expression: HasValueChanged(##DataPid:PID1.Sp) != nil

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.
 */

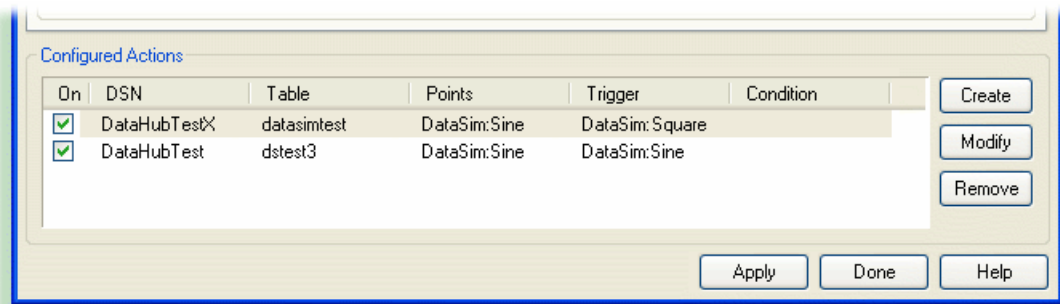
function HasValueChanged(point_symbol)
{
    local ptsym = symbol(point_symbol);
    local oldvalue = getprop(ptsym, #last_log_value);
    local curvalue;
    local result = nil;

    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;
}

```

9.9. 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.

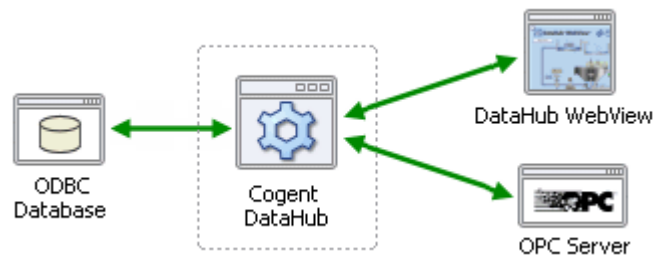
Notes

1. Text logging can be configured through DataHub scripting.

Chapter 10. Query a Database

10.1. 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 DataHub WebView.

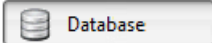


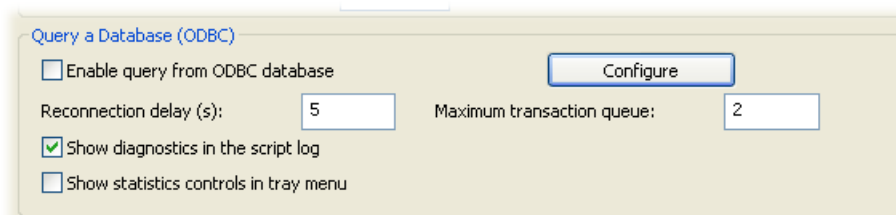
The Cogent DataHub's ODBC Data Logging interface provides an easy way to connect to a database and submit a query. There are two options for writing the query to the DataHub: one DataHub point per row, or the whole query results in a single DataHub point. The fastest way to learn how to use the interface is by using the [Quick Start](#).

10.2. Quick Start

Here's how to configure the Cogent DataHub to query a database of your choice.

Open the Query ODBC Database window

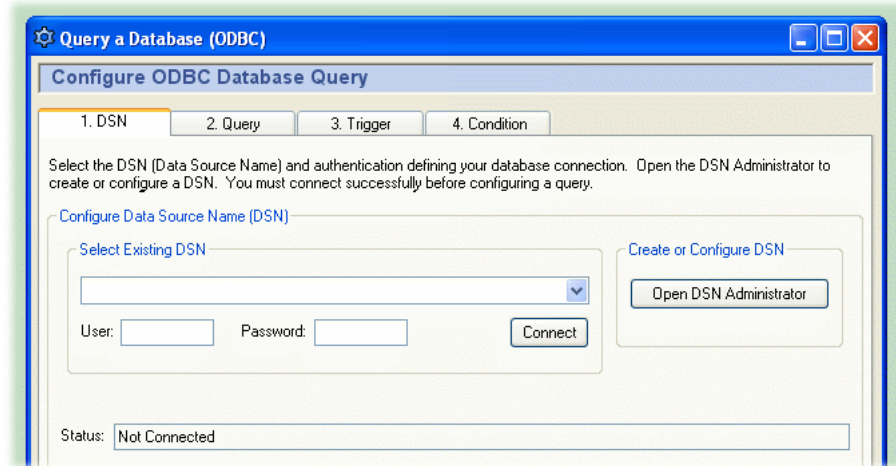
1. In the Cogent DataHub Properties window, select Database .
2. In the Query a Database (ODBC) section, click the Configure button.



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.



- 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.
- Enter the appropriate 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.

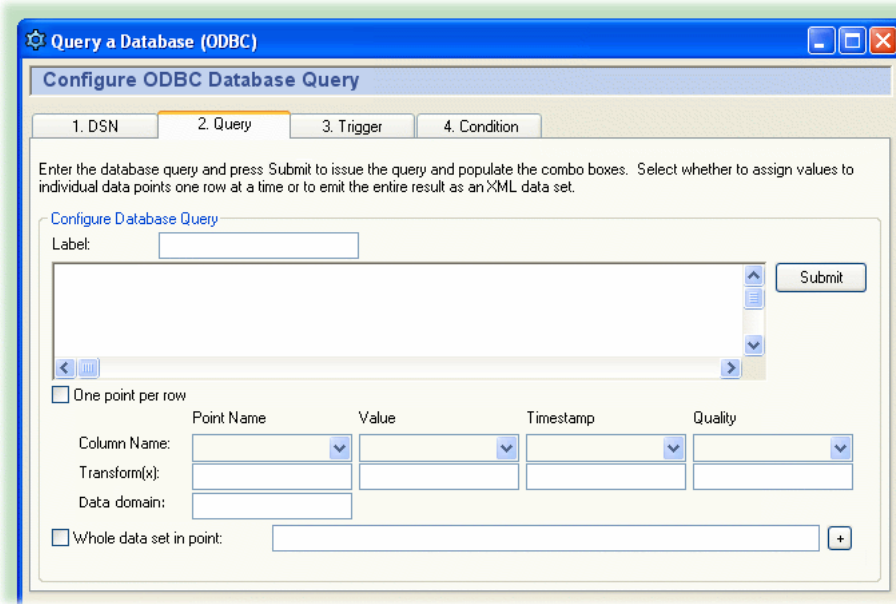
Configure a Query



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.

In our example, we will be writing the contents of the `VariableName` and `VariableValue` columns to the DataHub

- Select the **2. Query** tab.



- In the **Label:** field, enter a name for this query, such as **TestQuery**. This can be any string.
- Enter a valid SQL query. As an example, for our small database, we can use a simple SQL query:

```
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.

- 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.
- 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 [Section 10.4, Configuring a Database Query](#).

Select **One point per row**.

- 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`.
- 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`.
- 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`.
- Click the **Create** button.

On	Label	DSN	Query	Trigger
<input checked="" type="checkbox"/>	TestQuery	MySQLLocal	SELECT * FROM...	

A new configured action should appear in the list. For more information about configured actions, please refer to [Section 10.7, Configured Actions](#)

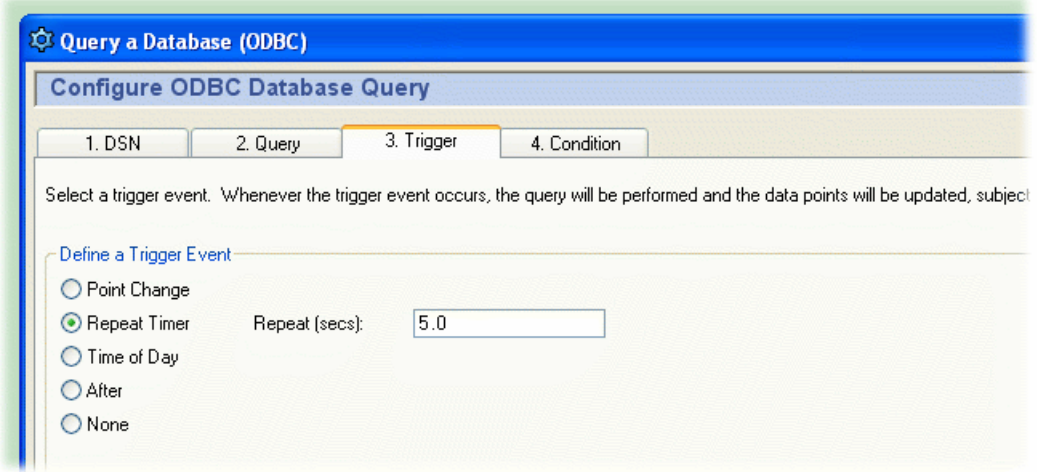
Next, to get the DataHub to make the query, you need to assign a trigger.

Assign a Trigger

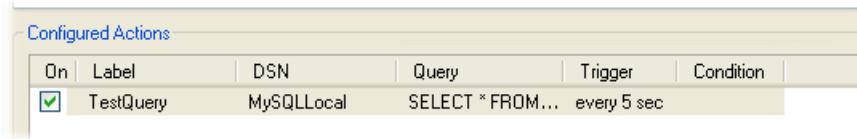
For this example, we will set a repeat timer to requery the database every 5 seconds.

- Select the **3. Trigger** tab.

2. Select Repeat Timer and enter 5.

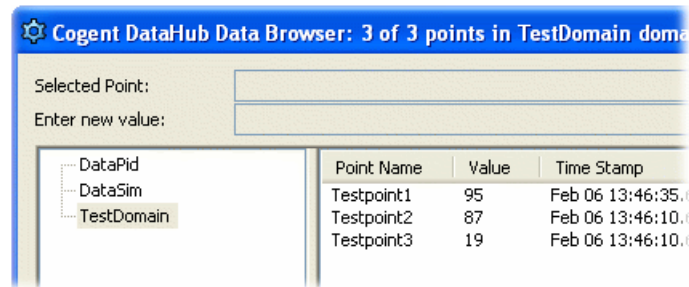


3. In the Configured Actions box, make sure that the configured action you just created is highlighted. If not, click on it to highlight it. Then click the Modify button.



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:



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.

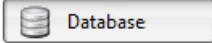
Should you not see the data points, or changes made to the values, you can open the [Script Log](#) to check for error messages and ensure that your query is being sent every 5 seconds.

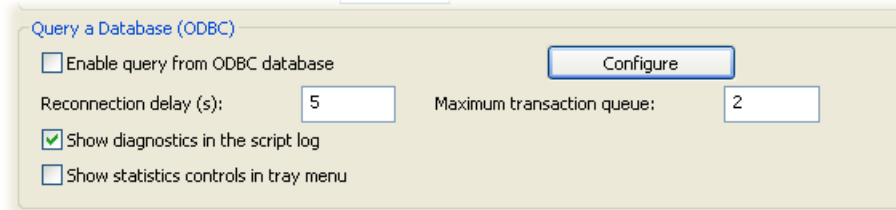
If all is working as described, you have configured an action that queries a database for and writes the results into the DataHub. The remaining sections in this chapter explain the interface in more detail, and introduce the option of setting specific [conditions](#) for queries, if desired.

10.3. Setting up the DSN (Data Source Name)

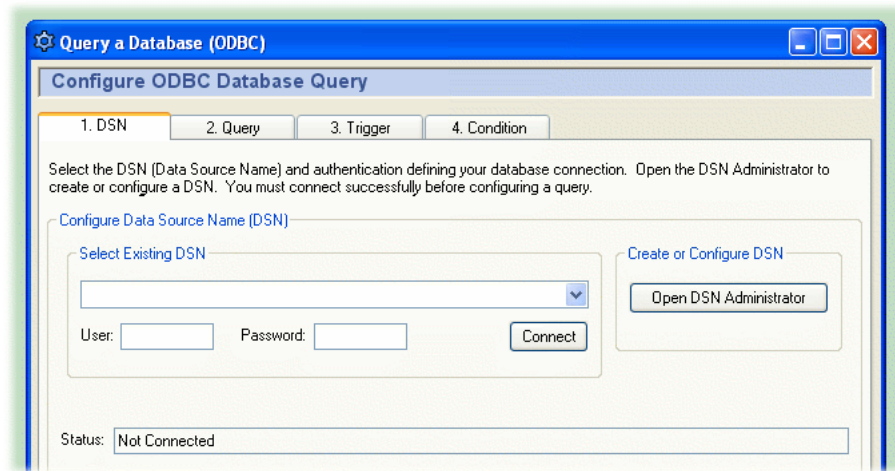
A *DSN* is a Data Source Name. Windows uses this name to identify the database you want to connect to. This tab lets you select an existing DSN, or create a new one if necessary.

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.



Selecting a DSN

1. To select a DSN, choose one from the drop-down box, and then enter the appropriate user name and password.



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.

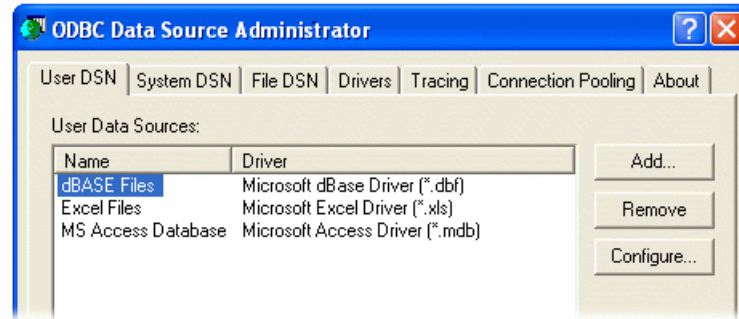
Creating or Configuring a DSN



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.

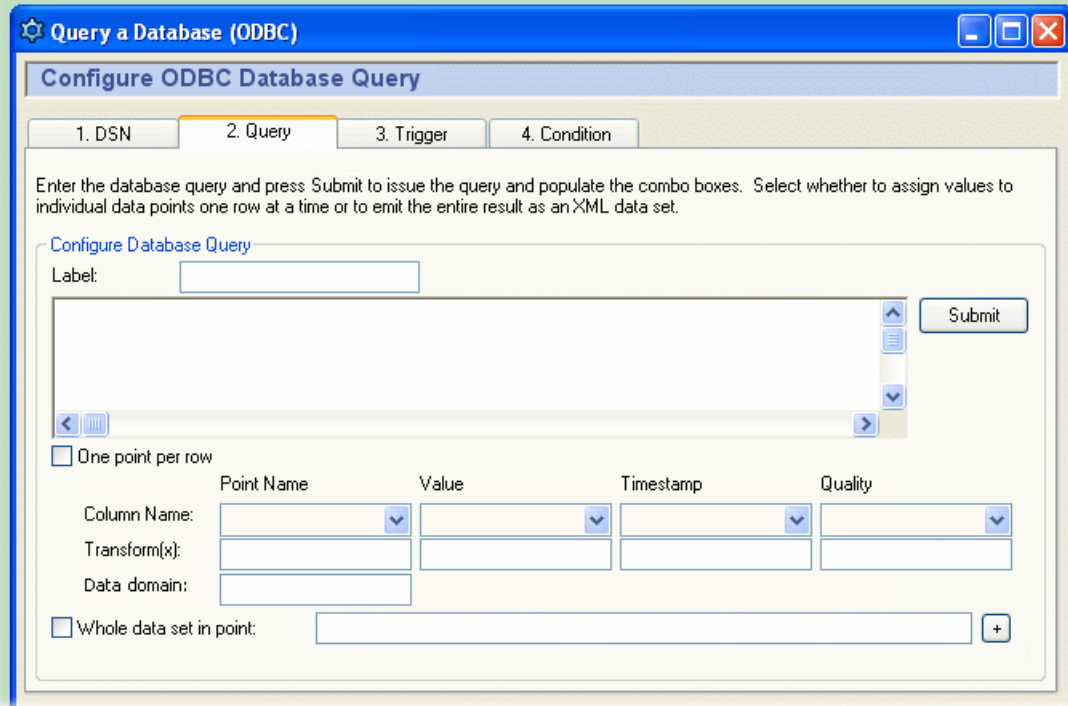
Configure an existing database

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.

10.4. Configuring a Database Query

After you have [set up a DSN](#) you can create a database query.

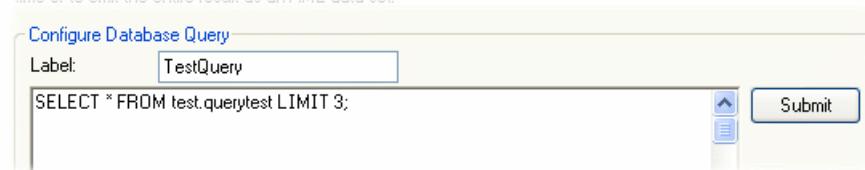


Preparing the Query



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.

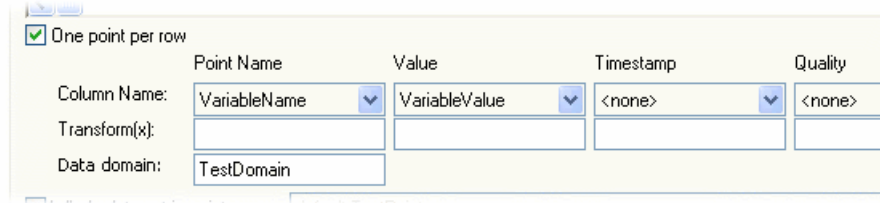
- In the **Label:** field, enter a name for this query, such as **TestQuery**. This can be any string.
- Enter a valid SQL query.



- 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

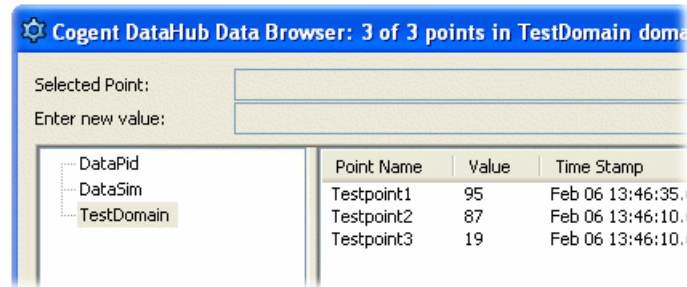
- The **One point per row** option lets you assign one point in the DataHub for each uniquely named item in one column of a database.



- 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.

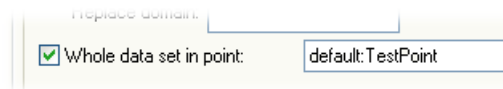
For example, an entry of `string("MyTag" , x)` under the Point Name would change points labelled Pump1 and Pump2 to MyTagPump1 and MyTagPump2. Or, an entry of `10 * x` under the Value would multiply each value by 10.

- 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

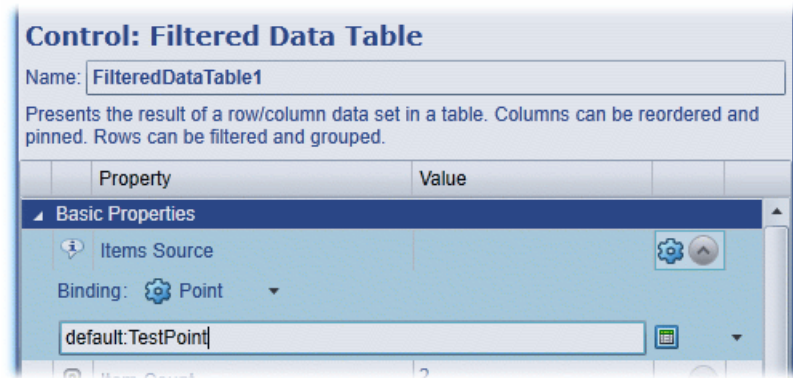
The Whole data set in point option lets you put the entire results of the query into one DataHub point, as specified, formatted in XML.



For example, our query of `SELECT * FROM test.querytest LIMIT 3;` on our example database would write the following XML-formatted data as the value of a single point:

```
<TableData>
  <column name="ID" type="int" datatype="-6" />
  <column name="VariableName" type="string" datatype="-9" />
  <column name="VariableValue" type="double" datatype="8" />
  <column name="Notes" type="string" datatype="-9" />
  <row>
    <ID>1</ID>
    <VariableName>Testpoint1</VariableName>
    <VariableValue>95</VariableValue>
    <Notes>First test point</Notes>
  </row>
  <row>
    <ID>2</ID>
    <VariableName>Testpoint2</VariableName>
    <VariableValue>87</VariableValue>
    <Notes>Second test point</Notes>
  </row>
  <row>
    <ID>3</ID>
    <VariableName>Testpoint3</VariableName>
    <VariableValue>19</VariableValue>
    <Notes>Third test point</Notes>
  </row>
</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:



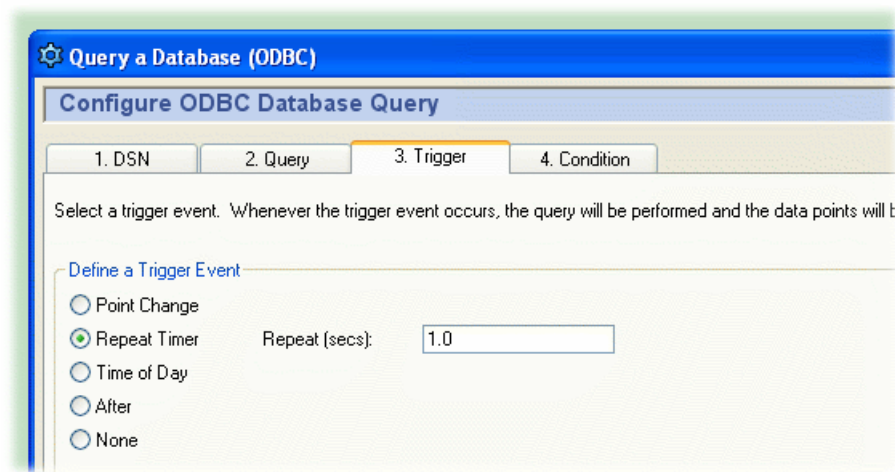
The data would then appear in the control:

ID	VariableName	VariableValue	Notes
1	Testpoint1	95	First test point
2	Testpoint2	87	Second test point
3	Testpoint3	19	Third test point

Page 1 of 1

10.5. 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 [Section 10.6, 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.. 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 [Section 10.6, Setting Trigger Conditions](#).

The ranges of the fields are:

Year:	1970 - *	Hour:	0 - 23
Month:	1 - 12	Minute:	0 - 59
Day:	1 - 31	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:

Year:	<input type="text" value="*"/>	Hour:	<input type="text" value="8"/>
Month:	<input type="text" value="*"/>	Minute:	<input type="text" value="45"/>
Day:	<input type="text" value="*"/>	Second:	<input type="text" value="0"/>

would trigger a query at 8:45 every day, every month, and every year.

- These entries:

Year:	*	Hour:	*
Month:	*	Minute:	0
Day:	15	Second:	0

would trigger a query every hour on the 15th day of each month, every year.

- These entries:

Year:	*	Hour:	8,10,12,14,16,18
Month:	*	Minute:	0-4
Day:	*	Second:	0

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.

10.6. 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:

When the trigger occurs, perform the query only if this condition is true

<input checked="" type="checkbox"/>		<input type="text" value="\$DataSim:Sine"/>	+	>	<input type="text" value="0"/>
<input checked="" type="checkbox"/>	And	<input type="text" value="thistrigger"/>	+	>	<input type="text" value="lasttrigger"/>
<input type="checkbox"/>	And	<input type="text"/>	+	==	<input type="text"/>
<input type="checkbox"/>	And	<input type="text"/>	+	==	<input type="text"/>

Expression: `($DataSim:Sine > 0 && thistrigger > lasttrigger)`

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:

When the trigger occurs, perform the query only if this condition is true

<input checked="" type="checkbox"/>		<input type="text" value="localtime(clock()).wday"/>	+	>	<input type="text" value="0"/>
<input checked="" type="checkbox"/>	And	<input type="text" value="localtime(clock()).wday"/>	+	<	<input type="text" value="6"/>

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.
.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 [Section 10.5, 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.

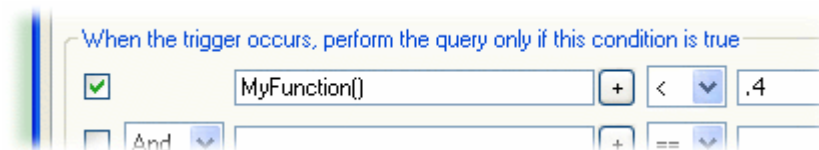
Custom Conditions

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 ()
{
    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 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
 */

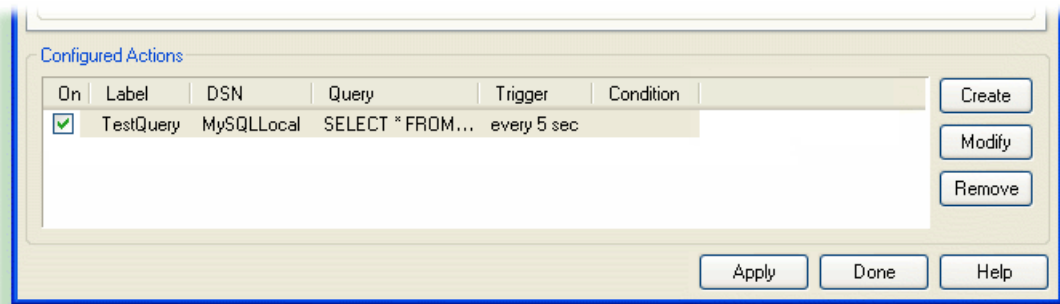
function HighWithReset(!triggerpoint)
{
    local value;
    if (!undefined_p(value = eval(triggerpoint)) && value != 0)
    {
        after(0.001, `setq(@triggerpoint, 0));
        t;
    }
    else
    {
        nil;
    }
}

/*
 * 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;
    }
}
```

10.7. 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.

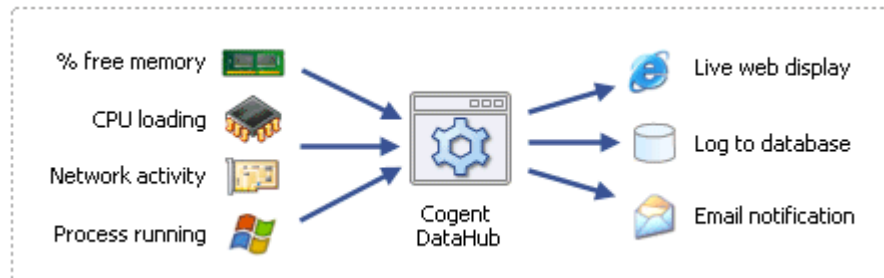
The Remove button removes a configured action.

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.

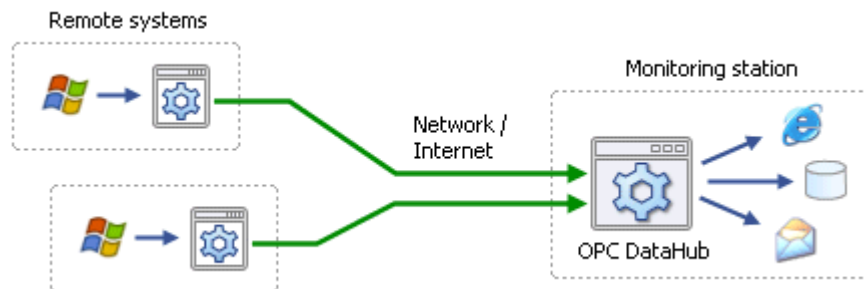
Chapter 11. System Monitor

11.1. 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.



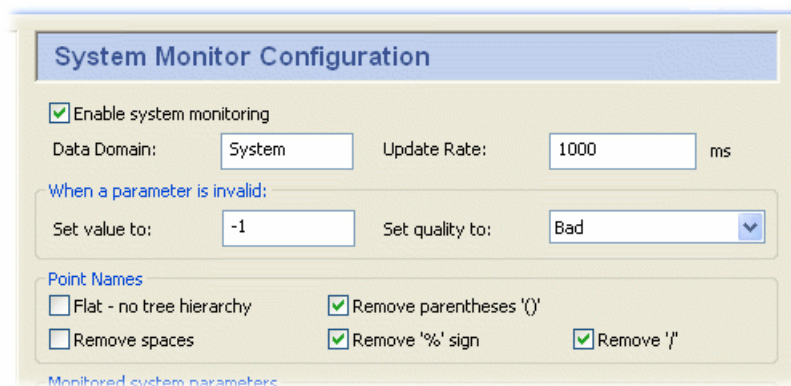
For example, by monitoring process ID you could determine whether a particular process is running or not. Any information accessed here becomes part of the DataHub's data set, and can thus be tunneled across the network, used in scripts or as email triggers, viewed in a spreadsheet, or stored in a database.



11.2. 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.

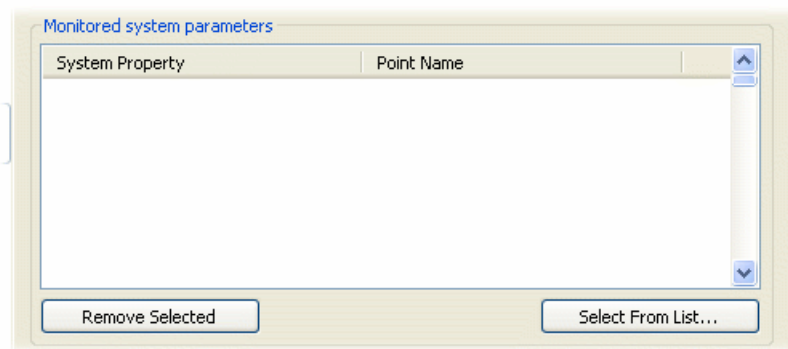
When a parameter is invalid:

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.

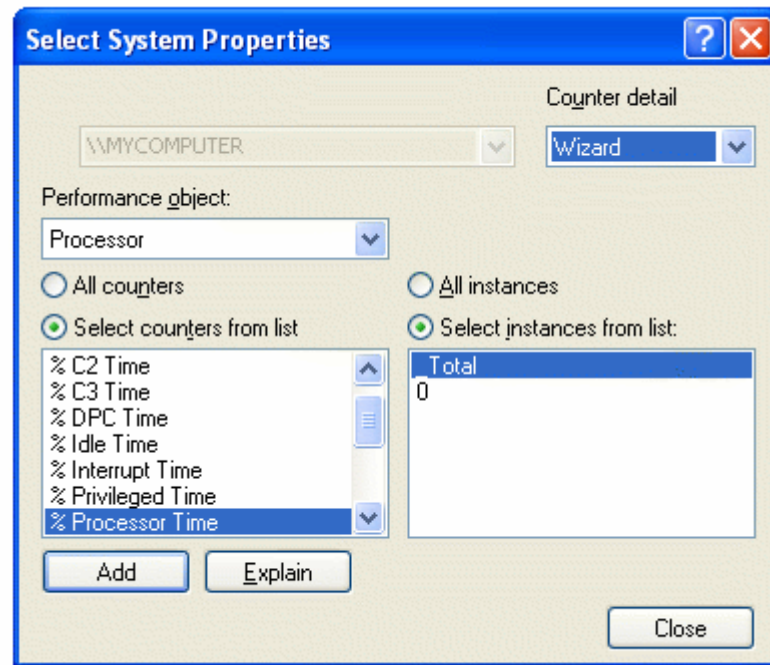
Point Names:

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:



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.

- 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 [Section 18.1, Data Points](#) for more information on creating and deleting points.

11.3. 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.

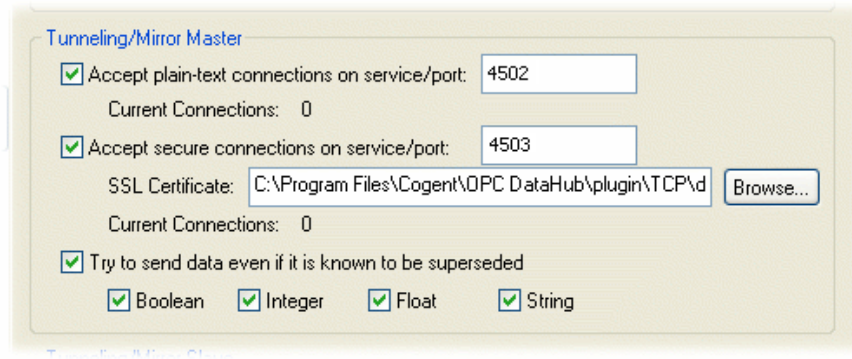


Mirroring is the same as *tunnelling*, as described in [Chapter 3, OPC Tunnelling](#). Mirroring can be also used to connect to Cascade DataHub running in Linux. Please refer to the Mirroring Data section of the Cascade DataHub for Linux and QNX manual for details.

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

- Right click on the Cogent DataHub system-tray icon and choose **Properties**.
- In the Properties window, select **Tunnel/Mirror** .




- 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 [Section 20.4, Tunnel/Mirror](#) for more information.

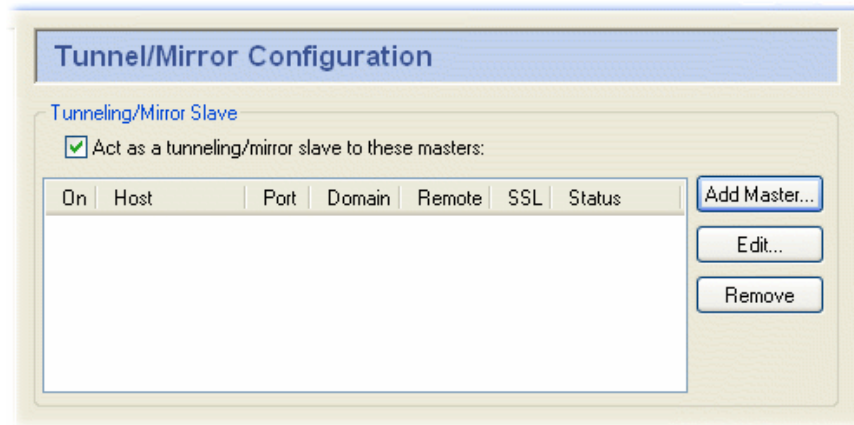


To optimize throughput, un-check the **Try to send data even if it is known to be superseded** option. This will allow the DataHub to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

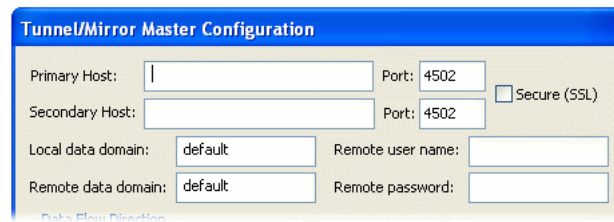
- Click **OK** to close the Properties window.
- You are now ready to configure the slave DataHub.

Configure the Cogent DataHub as tunnel/mirror slave

1. Right click on the Cogent DataHub system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror  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 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.

- **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.



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

- **Primary Host:** `developers.cogentrts.com`
- **Port:** 4502
- **Local data domain:** `test`
- **Remote data domain:** `test`

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 [Section 19.1, Binary Mode Tunnel/Mirror \(TCP\) Connections](#).

- **Target is a Cogent 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 for slow networks, please refer to [Section 19.2, Tunnel/Mirror \(TCP\) Connections for Slow Networks](#).

- **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.

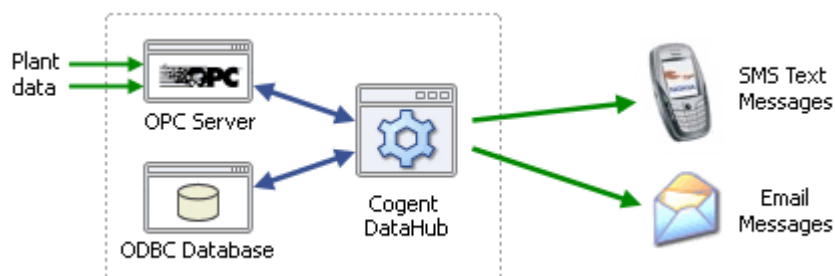
8. Click the **Apply** button in the Properties Window. If the master DataHub is running, this DataHub should establish the tunnelling connection, and the **Status** should display **Connected**. You can view the data with the [Data Browser](#), or view the connection with the [Connection Viewer](#).

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.

Chapter 12. Email and SMS

12.1. 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.

12.1.1. 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 message.](#)
3. [Send your own messages.](#)
4. [Create HTML messages.](#)

12.1.2. A note about SMS text messages

Sending SMS text messages from the DataHub is done by simply sending a small plain text email 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 subscribed with Bell Mobility in Canada, and your new cell phone number was (416) 123 4567, then the email address for this phone would be:

`4161234567@bellmobility.ca`

and the SMS text message address would be slightly different:

`4161234567@txt.bellmobility.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 regular email and it will be converted by the SMS gateway (`txt.bellmobility.ca` in the example) 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.

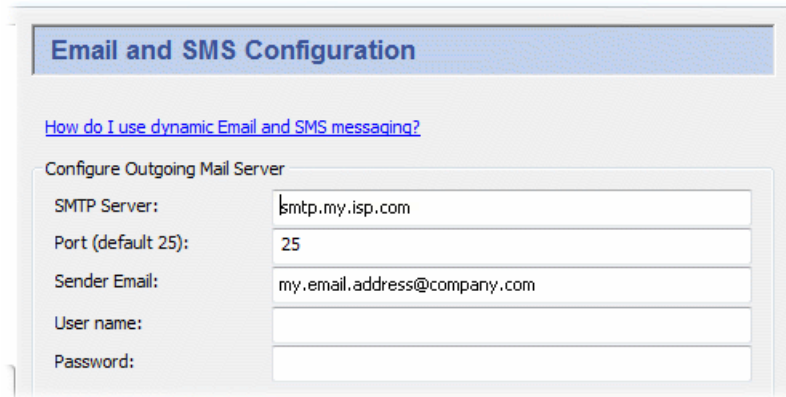
Check with your cell phone service provider for the SMS text message address for your cell phone number.

12.2. 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**  **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.

3. In the **Security** section:



Choose one of the three SSL options, and specify whether you want to accept invalid or untrusted security certificates.

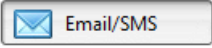
4. Click the **Apply** or **OK** button to submit your entries.

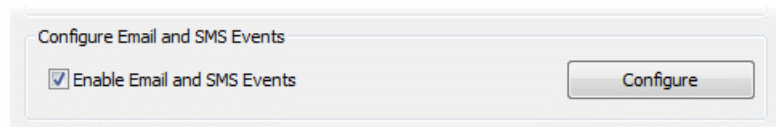
The DataHub mailer is now ready to use. If you haven't already done so, we suggest [sending a test message](#) as explained in the next section.

12.3. Sending a Test Message

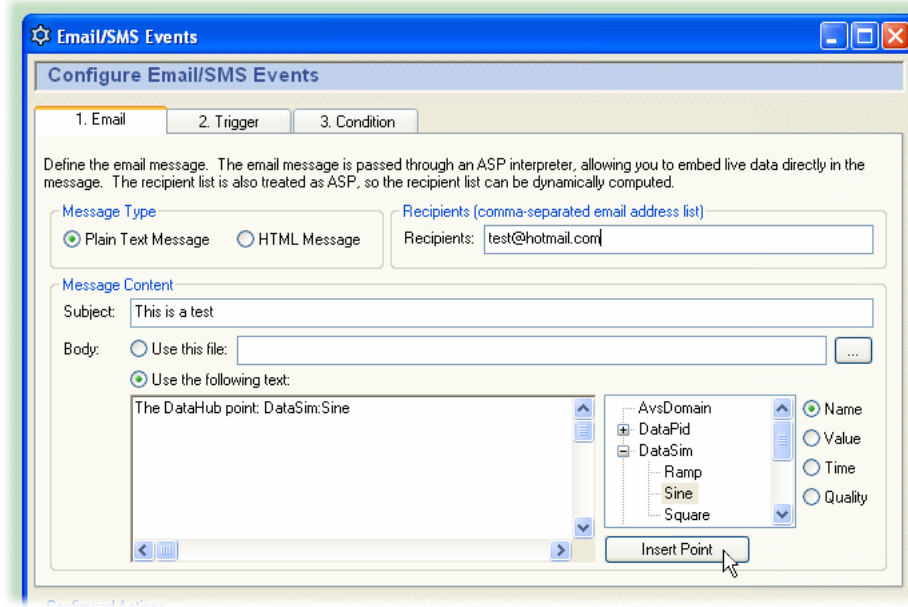
Once you have [configured the mail server](#) you can configure and send a test email. Here's how:

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 DataHub point
```
10. Click the Insert Point button. Your text display should now look like this:

```
The DataHub point DataSim:Sine
```
11. Press **Enter** and continue typing:

```
The DataHub 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 DataHub point DataSim:Sine
had a value of: <%= $DataSim:Sine %>
```
13. Press **Enter** and continue typing:

```
The DataHub 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:

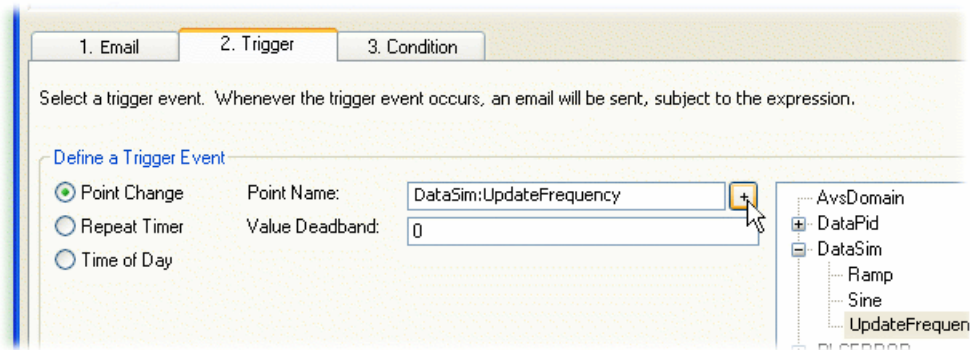
```
The DataHub point DataSim:Sine
had a value of: <%=DataSim:Sine%>
at the time: <%=PointTimeString(#$DataSim:Sine%)>
```

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.

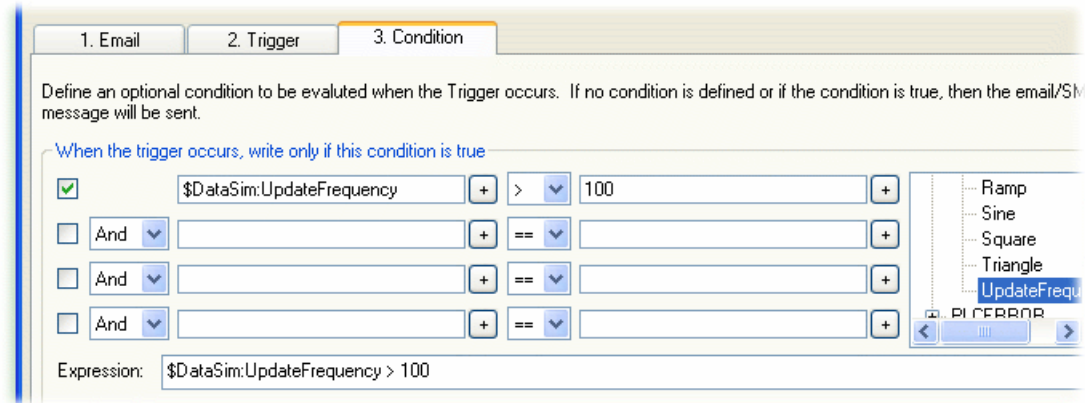


You can choose any point for the trigger, including the point that gets written, such as DataSim:Sine in our example. For more information about triggers, please refer to [Section 12.5, Assigning a Trigger](#).

Set a Condition and Configure the Action

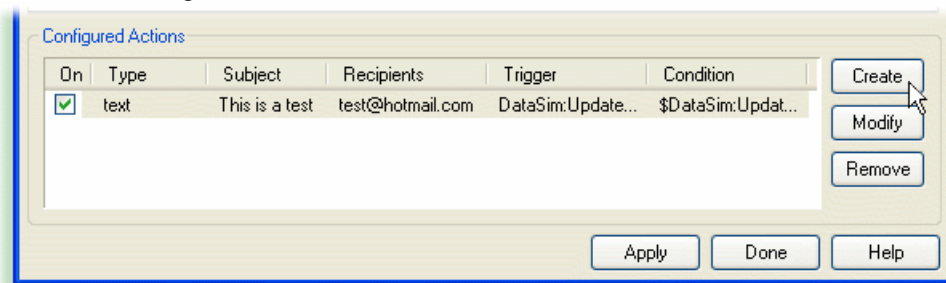
For this example, let's limit the trigger on the DataSim:UpdateFrequency point to changes only to values over 100.

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:



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.



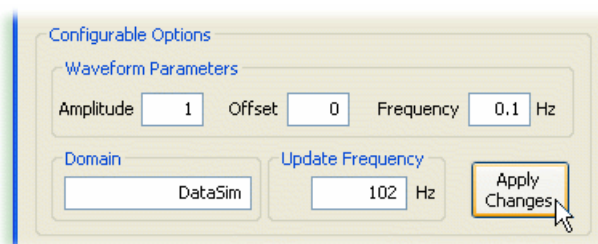
A new configured action should appear in the list. This is a summary of what you have done. When a configured action is selected in this list, you can make changes in any of the tabs and modify it using the **Modify**. You can also duplicate a configured action using the **Create** button, or remove it with the **Remove** button. For more information about configured actions, please refer to [Section 12.7, Configured Actions](#)

8. Click the **Apply** button to activate the configured action. Now let's see how it all works.

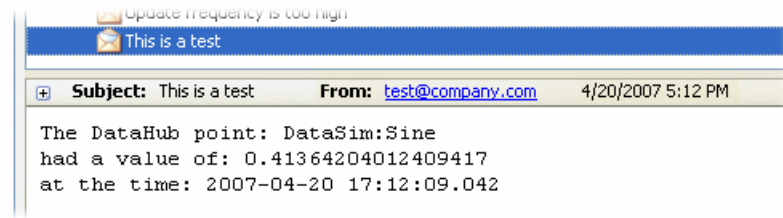
Trigger and Send an Email

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.



3. Check the email account of the recipient. You should have received a message that looks like this:




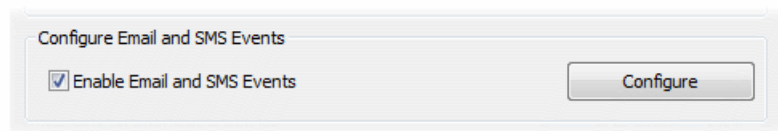
Each time you enter a new Update Frequency value greater than 100 in the DataSim, the DataHub script will send a similar message.

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

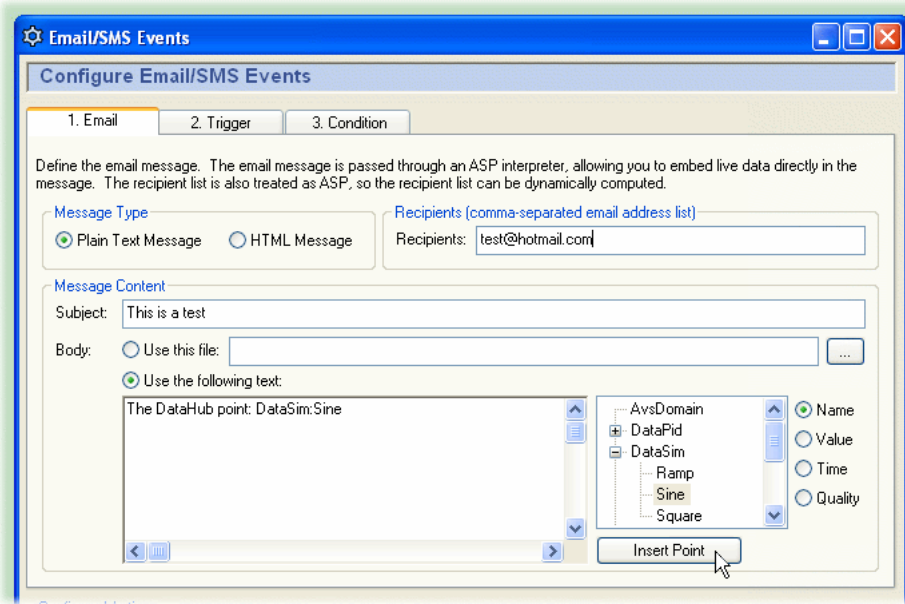
12.4. Defining the Email Message

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  Email/SMS .
2. In the Configure Email section press the Configure button.



3. Select the 1. Email tab and configure your email as explained below.



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 [Section 12.9, 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 [Section 12.8, 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:

Button	Syntax	Example
Name	<code>domainname:pointname</code>	DataSim:Sine
Value	<code><%= \$domainname:pointname %></code>	<code><%= \$DataSim:Sine %></code>
Time	<code><%= PointTimeString (\$domainname:pointname) %></code>	<code><%= PointTimeString (\$DataSim:Sine) %></code>
Quality	<code><%= PointQualityString (\$domainname:pointname) %></code>	<code><%= PointQualityString (\$DataSim:Sine) %></code>

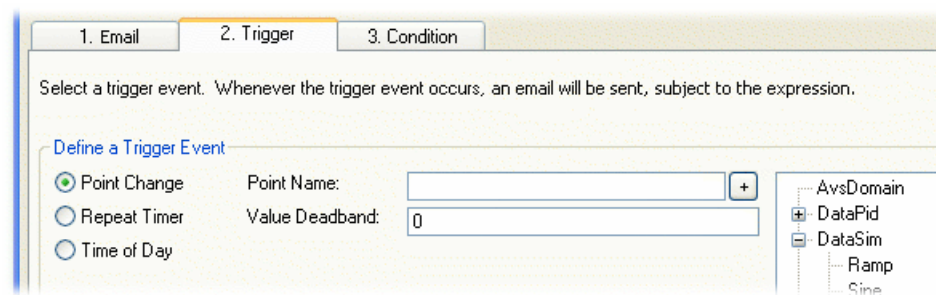
In this syntax, the special characters are used as follows:

Character	Use
-----------	-----

Character	Use
<% ... %>	The enclosed expression will be evaluated by Gamma, the DataHub scripting language.
\$	Indicates to Gamma that this is a DataHub point name.
PointTimeString()	A Gamma function that returns the timestamp of a DataHub point in an easily readable format.
PointQualityString()	A Gamma function that returns the quality of a DataHub point, as a text string.
#	Protects the DataHub point from being evaluated by Gamma until the function is called.

12.5. 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 [Section 12.6, 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 [Section 12.6, Setting Trigger Conditions](#).

The ranges of the fields are:

Year:	1970–*	Hour:	0–23
Month:	1–12	Minute:	0–59
Day:	1–31	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:

Year:	<input type="text" value="*"/>	Hour:	<input type="text" value="8"/>
Month:	<input type="text" value="*"/>	Minute:	<input type="text" value="45"/>
Day:	<input type="text" value="*"/>	Second:	<input type="text" value="0"/>

would cause an email to be sent at 8:45 every day, every month, and every year.

- These entries:

Year:	<input type="text" value="*"/>	Hour:	<input type="text" value="*"/>
Month:	<input type="text" value="*"/>	Minute:	<input type="text" value="0"/>
Day:	<input type="text" value="15"/>	Second:	<input type="text" value="0"/>

would cause an email to be sent every hour on the 15th day of each month, every year.

- These entries:

Year:	<input type="text" value="*"/>	Hour:	<input type="text" value="8,10,12,14,16,18"/>
Month:	<input type="text" value="*"/>	Minute:	<input type="text" value="0-4"/>
Day:	<input type="text" value="*"/>	Second:	<input type="text" value="0"/>

would cause an email to be sent every second for 5 minutes, every two hours between 8 a.m. and 6 p.m.

12.6. 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:

When the trigger occurs, write only if this condition is true

☒

☒ And

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:

<code>.sec</code>	The number of seconds after the minute (0 - 59).
<code>.min</code>	The number of minutes after the hour (0 - 59).
<code>.hour</code>	The number of hours past midnight (0 - 23).
<code>.mday</code>	The day of the month (1 - 31).
<code>.mon</code>	The number of months since January (0 - 11)
<code>.year</code>	The number of years since 1900.
<code>.wday</code>	The number of days since Sunday (0 - 6).
<code>.yday</code>	The number of days since January 1 (0 - 365)
<code>.isdst</code>	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 [Section 12.5, 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.

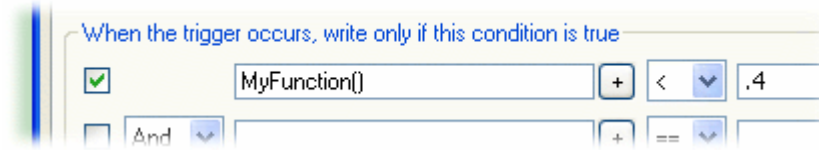
Custom Conditions

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`:

```
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)!= 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)
  {

```

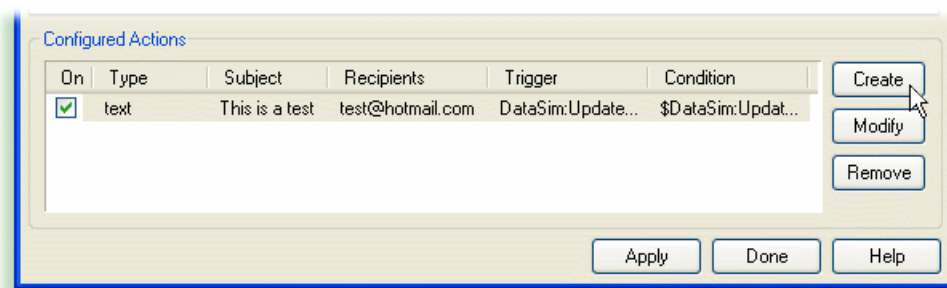
```

        after(0.001, 'setq(@triggerpoint, 1));
    t;
}
else
{
    nil;
}
}

```

12.7. 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.

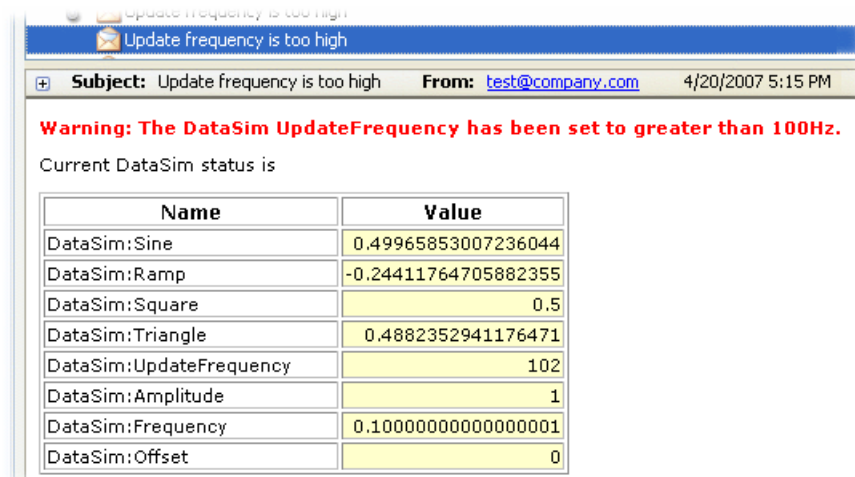
Once a configured action has been created or modified, the changes won't take effect until you click the **Apply** or **Done** button.

12.8. 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

12.8.1. 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:



Contents of the ASP File

The contents of the file `EmbedPoints.asp` are as follows:

```
<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: #23cce6fe;
}
.highlight{background-color: #FFFFCC; text-align:right;}
.warning{color: #FF0000; font-weight: bold;}
</style>
<body>
<!--
This is a simple example of an HTML template file which contains
embedded point values from the DataHub.
-->
<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>
</tr>
<tr>
<td>DataSim:Sine</td>
<td class="highlight"><%= $DataSim:Sine%></td>
</tr>
<tr>
<td>DataSim:Ramp</td>
<td class="highlight"><%= $DataSim:Ramp%></td>
</tr>
<tr>
<td>DataSim:Square</td>
<td class="highlight"><%= $DataSim:Square%></td>
</tr>
<tr>
<td>DataSim:Triangle</td>
<td class="highlight"><%= $DataSim:Triangle%></td>
</tr>
<tr>
<td>DataSim:UpdateFrequency</td>
<td class="highlight"><%= $DataSim:UpdateFrequency%></td>
</tr>
<tr>
<td>DataSim:Amplitude</td>
<td class="highlight"><%= $DataSim:Amplitude%></td>
</tr>
<tr>
<td>DataSim:Frequency</td>
<td class="highlight"><%= $DataSim:Frequency%></td>
</tr>
<tr>
<td>DataSim:Offset</td>
<td class="highlight"><%= $DataSim:Offset%></td>
</tr>
</table>
</body>
</html>
```



```

        <td>DataSim:Triangle</td>
        <td class="highlight"><%= $DataSim:Triangle%></td>
    </tr>
    <tr>
        <td>DataSim:UpdateFrequency</td>
        <td class="highlight"><%= $DataSim:UpdateFrequency%></td>
    </tr>
    <tr>
        <td>DataSim:Amplitude</td>
        <td class="highlight"><%= $DataSim:Amplitude%></td>
    </tr>
    <tr>
        <td>DataSim:Frequency</td>
        <td class="highlight"><%= $DataSim:Frequency%></td>
    </tr>
    <tr>
        <td>DataSim:Offset</td>
        <td class="highlight"><%= $DataSim:Offset%></td>
    </tr>
</table>
</body>
</html>

```

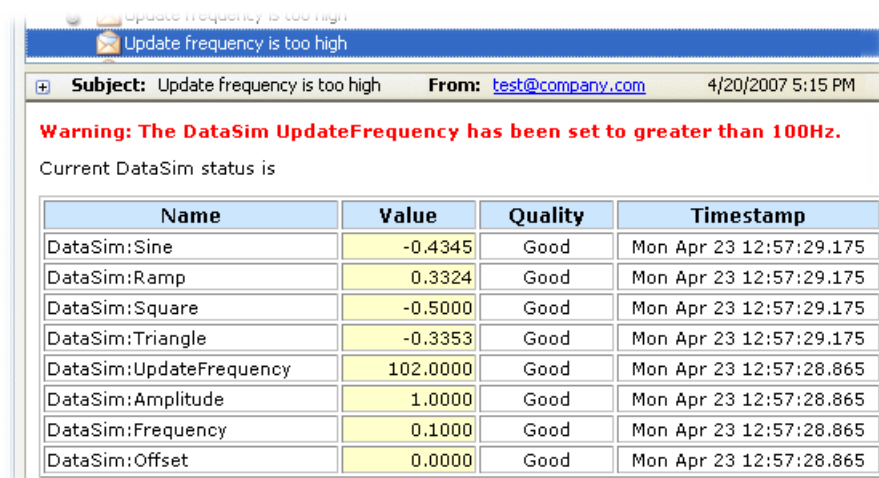
This file consists of HTML code interspersed with *Gamma* code. Gamma is the scripting language of the Cogent DataHub. The Gamma code is often used to determine the value of a DataHub point, with the following syntax:

```
<%= $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.

12.8.2. 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:



Warning: The DataSim UpdateFrequency has been set to greater than 100Hz.

Current DataSim status is

Name	Value	Quality	Timestamp
DataSim:Sine	-0.4345	Good	Mon Apr 23 12:57:29.175
DataSim:Ramp	0.3324	Good	Mon Apr 23 12:57:29.175
DataSim:Square	-0.5000	Good	Mon Apr 23 12:57:29.175
DataSim:Triangle	-0.3353	Good	Mon Apr 23 12:57:29.175
DataSim:UpdateFrequency	102.0000	Good	Mon Apr 23 12:57:28.865
DataSim:Amplitude	1.0000	Good	Mon Apr 23 12:57:28.865
DataSim:Frequency	0.1000	Good	Mon Apr 23 12:57:28.865
DataSim:Offset	0.0000	Good	Mon Apr 23 12:57:28.865

Contents of the ASP File

The contents of the file `CreateTable.asp` are as follows:

```
<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);
        v = eval (pt);
        if (!number_p(v)) v = 0;
        q = GetQualityName (info.quality);
        ts = PointGetUnixTime (pt);
        tm = format ("%19.19s.%03d", date(ts), (ts % 1.0) * 1000);
    %>
        <tr><td><%= pt %></td>
            <td class="highlight"><%= format("%.4f",v) %></td>
            <td align="center"><%= q %></td>
            <td align="center"><%= tm %></td></tr>
    <%
    }
}
catch
{
    princ (_last_error_, "\n");
    print_stack (nil, _error_stack_);
}
%>
</table>
</body>
</html>
```

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.

12.9. 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 end of [Section 12.4, 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 could 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:

```
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:

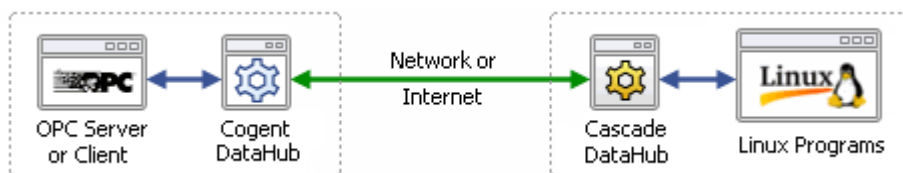
```
x = insert_expression_here;
```

Chapter 13. Linux Connections

You can access live data in Linux, and bring data into the Cogent DataHub from any Linux program, using the Cogent DataHub in Windows and the Cascade DataHub in Linux. Once connected, the two DataHubs mirror data across a network or the Internet.



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.



There are two steps to set this up:

1. [Configure the Cogent DataHub on the Windows machine.](#)
2. [Configure the Cascade DataHub on the Linux machine.](#)


13.1. Configuring the Cogent DataHub in Windows

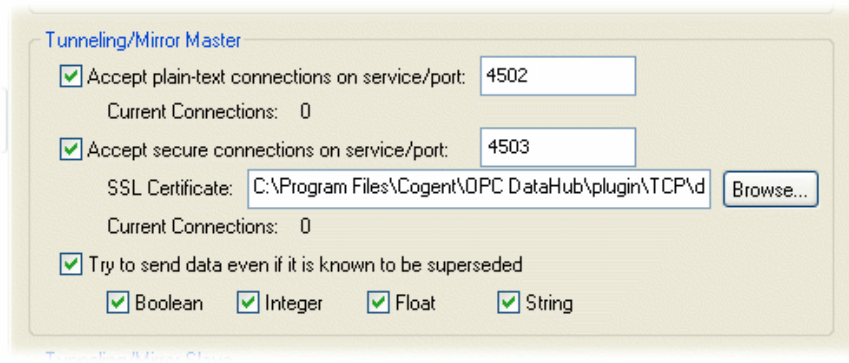
1. Connect the DataHub to the OPC server or the OPC client, if you haven't already done so. Please refer to [Section 2.3, Connect to an OPC server](#) or [Section 2.4, Connect from an OPC client](#) for instructions on how to do this.
2. Configure the DataHub for tunnelling. To do this, you must decide which DataHub (on Windows or on Linux) will be the master, and which the slave. The master DataHub receives the initial request from a slave to establish the connection, initially or after a network break. For this reason, we suggest that the computer with the most expected up time be the master. This could be either the Windows or the Linux machine, depending on your individual circumstances.



If the Windows computer is going to act as the master, follow the first procedure. Otherwise, follow the second procedure.

Configure the Cogent DataHub as tunnelling 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 [Section 20.4, Tunnel/Mirror](#) for more information.

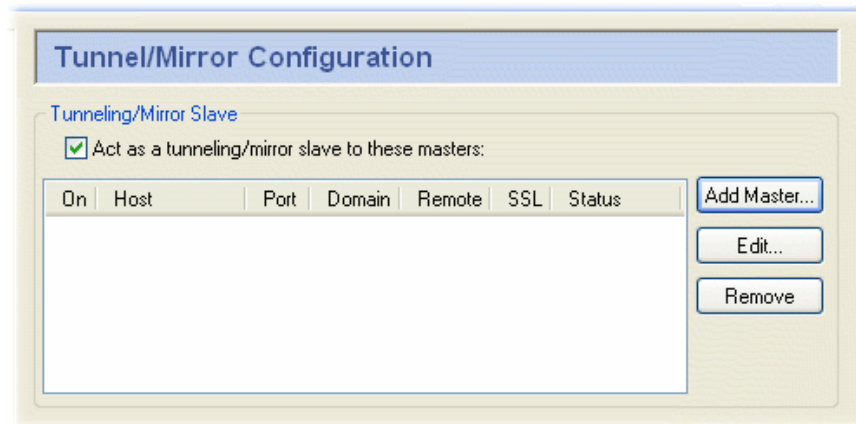


To optimize throughput, un-check the Try to send data even if it is known to be superseded option. This will allow the DataHub to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

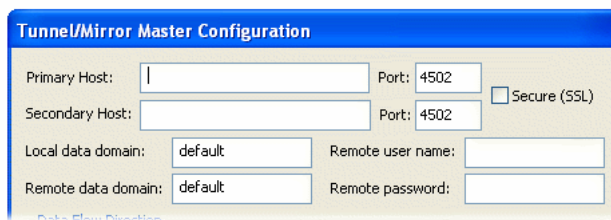
4. Click OK to close the Properties window.

Configure the Cogent DataHub as tunnelling slave

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 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.

- **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.



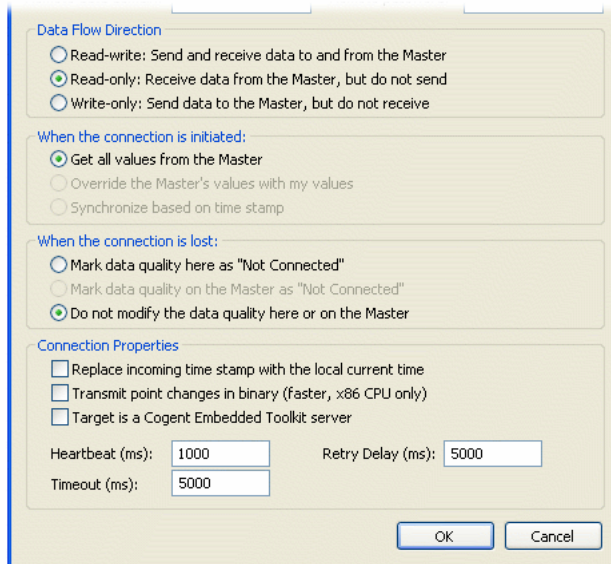
Unless you have a good reason for making these different, we recommend using the same data domain name on both DataHubs for the sake of simplicity.



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

- **Primary Host:** `developers.cogentrts.com`
- **Port:** 4502
- **Local data domain:** `test`
- **Remote data domain:** `test`

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 [Section 19.1, Binary Mode Tunnel/Mirror \(TCP\) Connections](#).

- **Target is a Cogent 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 for slow networks, please refer to [Section 19.2, Tunnel/Mirror \(TCP\) Connections for Slow Networks](#).

- **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](#).

Now you are ready to configure the Cascade DataHub on Linux.

13.2. Configuring the Cascade DataHub on Linux

You can configure the Cascade DataHub on Linux from the command line when you start the DataHub using the **datahub** command. A complete list of the **datahub** command options is available in the Cascade DataHub for Linux and QNX manual. Here we will explain only the necessary ones.

Configure the Cascade DataHub as mirroring slave

If the Cogent DataHub on the Windows machine has been configured as a tunnelling master, the Cascade DataHub on the Linux machine must be configured as a mirroring slave.

1. Start the DataHub using this command-line syntax:

```
[sh]$ datahub -M address -m port -n domain
```

where:

address

The name or IP address of the Windows computer.

port

The TCP port number used by the DataHub for mirroring. The DataHub's preconfigured default is 4502.

domain

The name of the data domain on the Cogent DataHub where the OPC data is coming from.

Once you enter this command, the Cascade DataHub will start on the Linux machine and attempt to connect to the Cogent DataHub on the Windows machine.

2. To see the data, start **xdhview** like this:


```
[sh]$ xdhview -d domain &
```

where *domain* is the same as you entered for the DataHub. When you enter this command, the X Windows version of a DataHub viewer should open, allowing you to see the incoming OPC data.

Configure the Cascade DataHub as mirroring master

If the Cogent DataHub on the Windows machine has been configured as a tunnelling slave, the Cascade DataHub on the Linux machine must be configured as a mirroring master.

1. Start the DataHub like this:

```
[sh]$ datahub -d domain -p 4502
```

where *domain* is the same name you entered on the Windows machine as the Remote data domain.

Once you enter this command, the Cascade DataHub will start on the Linux machine and should accept any attempt by the Cogent DataHub on the Windows machine to make a connection.

2. To see the data, start **xdhview** like this:

```
[sh]$ xdhview -d domain &
```

where *domain* is the same as you entered for the DataHub. When you enter this command, the X Windows version of a DataHub viewer should open, allowing you to see the incoming OPC data.

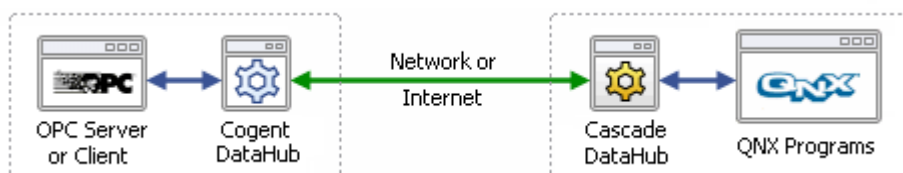
Please refer to the Cascade DataHub for Linux and QNX manual for more information about using the Cascade DataHub in Linux.

Chapter 14. QNX Connections

You can access live data in QNX 4 or QNX 6, and bring data into the Cogent DataHub from any QNX 4 or QNX 6 program, using the Cogent DataHub in Windows and the Cascade DataHub in QNX. Once connected, the two DataHubs mirror data across a network or the Internet.



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.



There are two steps to set this up:

1. [Configure the Cogent DataHub on the Windows machine.](#)
2. [Configure the Cascade DataHub on the QNX machine.](#)


14.1. Configuring the Cogent DataHub in Windows

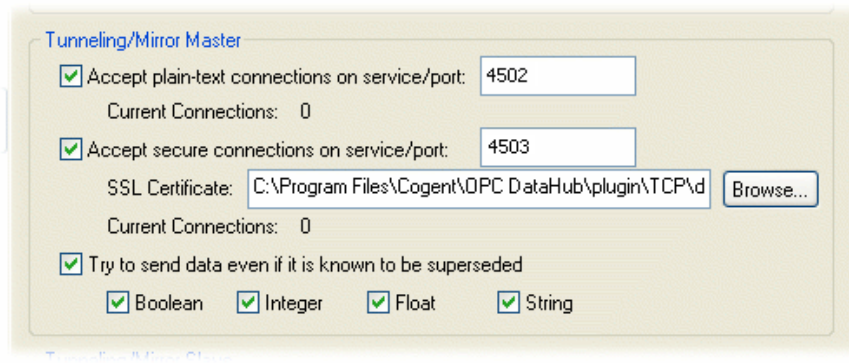
1. Connect the DataHub to the OPC server or the OPC client, if you haven't already done so. Please refer to [Section 2.3, Connect to an OPC server](#) or [Section 2.4, Connect from an OPC client](#) for instructions on how to do this.
2. Configure the Cogent DataHub for tunnelling. To do this, you must decide which DataHub (on Windows or on QNX) will be the master, and which the slave. The master DataHub receives the initial request from a slave to establish the tunnelling connection, initially or after a network break. For this reason, we suggest that the computer with the most expected up time be the master. This could be either the Windows or the QNX machine, depending on your individual circumstances.



If the Windows computer is going to act as the master, follow the first procedure. Otherwise, follow the second procedure.

Configure the Cogent DataHub as tunnelling 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 [Section 20.4, Tunnel/Mirror](#) for more information.

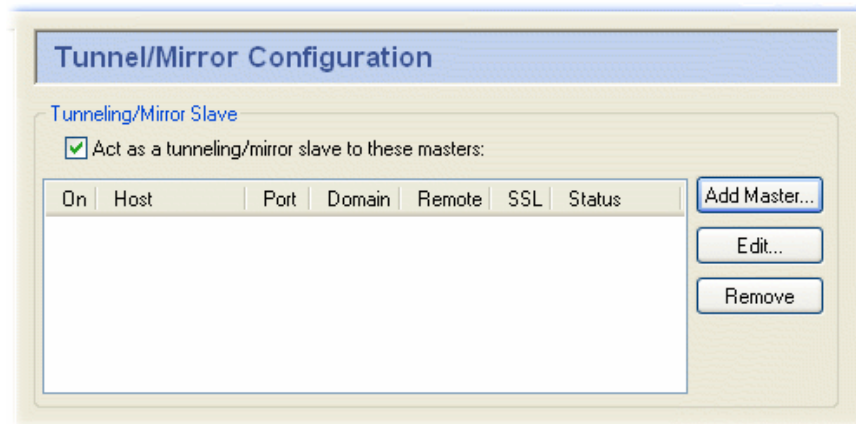


To optimize throughput, un-check the Try to send data even if it is known to be superseded option. This will allow the DataHub to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

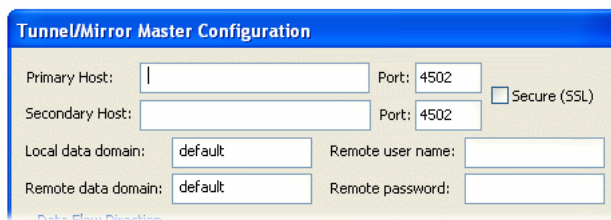
4. Click OK to close the Properties window.

Configure the Cogent DataHub as tunnelling slave

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 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.

- **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.



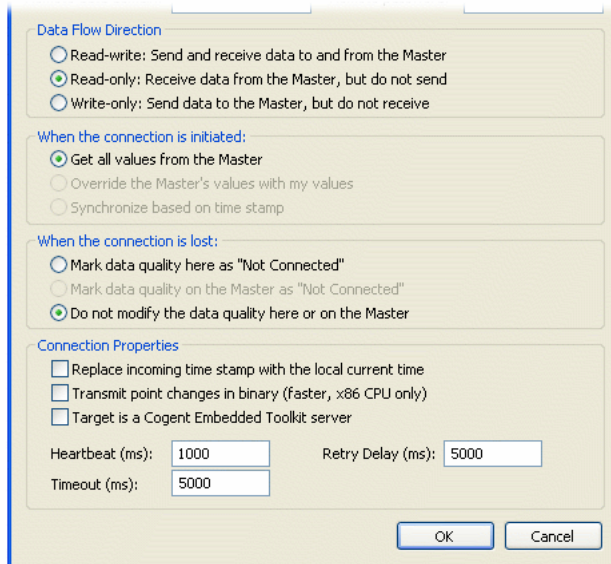
Unless you have a good reason for making these different, we recommend using the same data domain name on both DataHubs for the sake of simplicity.



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

- **Primary Host:** `developers.cogentrts.com`
- **Port:** 4502
- **Local data domain:** `test`
- **Remote data domain:** `test`

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 [Section 19.1, Binary Mode Tunnel/Mirror \(TCP\) Connections](#).

- **Target is a Cogent 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 for slow networks, please refer to [Section 19.2, Tunnel/Mirror \(TCP\) Connections for Slow Networks](#).

- **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](#).

Now you are ready to configure the Cascade DataHub on QNX.

14.2. Configuring the Cascade DataHub on QNX

You can configure the Cascade DataHub on QNX from the command line when you start the DataHub using the **datahub** command. A complete list of the **datahub** command options is available in the Cascade DataHub for Linux and QNX manual. Here we will explain only the necessary ones.

Configure the Cascade DataHub as mirroring slave

If the Cogent DataHub on the Windows machine has been configured as a tunnelling master, the Cascade DataHub on the QNX machine must be configured as a mirroring slave.

1. Start the DataHub using this command-line syntax:

```
[sh]$ datahub -M address -m port -n domain
```

where:

address

The name or IP address of the Windows computer.

port

The TCP port number used by the DataHub for mirroring. The DataHub's preconfigured default is 4502.

domain

The name of the data domain on the Cogent DataHub where the OPC data is coming from.

Once you enter this command, the Cascade DataHub will start on the QNX machine and attempt to connect to the Cogent DataHub on the Windows machine.

2. To see the data, start **phdhview** like this:

```
[sh]$ phdhview -d domain &
```

where *domain* is the same as you entered for the DataHub. When you enter this command, the X Windows version of a DataHub Viewer should open, allowing you to see the incoming OPC data.

Configure the Cascade DataHub as mirroring master

If the Cogent DataHub on the Windows machine has been configured as a tunnelling slave, the Cascade DataHub on the QNX machine must be configured as a mirroring master.

1. Start the DataHub like this:

```
[sh]$ datahub -d domain -p 4502
```

where *domain* is the same name you entered on the Windows machine as the Remote data domain.

Once you enter this command, the Cascade DataHub will start on the QNX machine and should accept any attempt by the Cogent DataHub on the Windows machine to make a connection.

2. To see the data, start **phdhview** like this:

```
[sh]$ phdhview -d domain &
```

where *domain* is the same as you entered for the DataHub. When you enter this command, the X Windows version of a DataHub Viewer should open, allowing you to see the incoming OPC data.

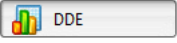
Please refer to the Cascade DataHub for Linux and QNX manual for more information about using the Cascade DataHub in QNX.

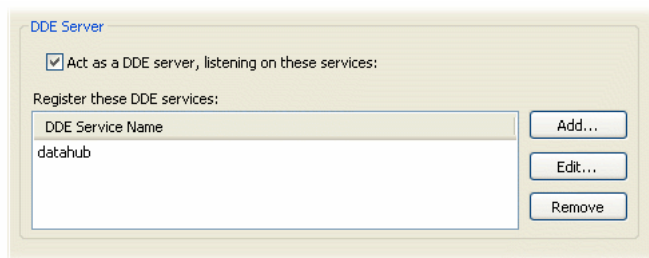
Chapter 15. InTouch Connections

15.1. Getting data into InTouch

The Cogent DataHub can act as a DDE server to your InTouch application. To access data from the Cogent DataHub you simply need to configure an InTouch Access Name which points to the cdh data domain containing the data you are interested in. Then you just associate tagnames in InTouch with corresponding cdh point names.

Configuring the Cogent DataHub

1. Right click on the Cogent DataHub system-tray icon and choose Properties.
2. In the Properties window, select DDE .

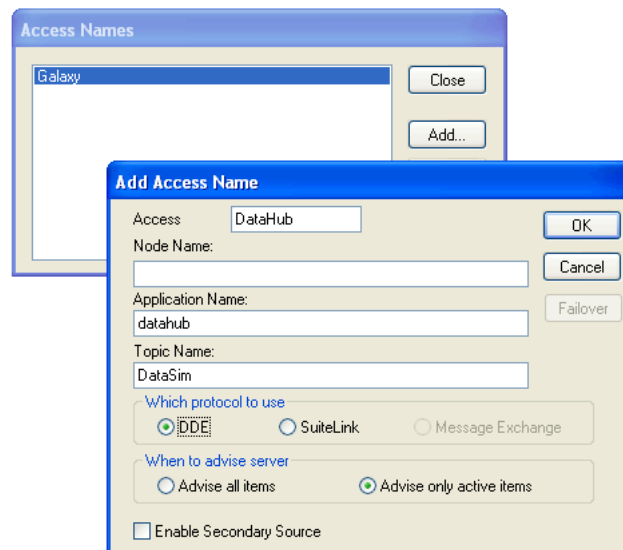


3. Ensure that the box Act as a DDE server is checked, and that the name datahub appears in the DDE Service Name area. If not, click the Add... button and add the name datahub.
4. Click OK to close the Properties window.

Configuring InTouch Access Names

For InTouch to establish the connection to the Cogent DataHub you need to set up an Access Name.

1. In the InTouch WindowMaker program, from the Special menu, select Access Names.
2. Click the Add button to add a new Access Name.



3. Enter the information in the Add Access Name window as follows:

Access

Create a name.

Node Name

Can be left blank if the DataHub is running on the same computer as the InTouch application.
Otherwise, the name of the node that the DataHub is running on.

Application Name

datahub

Topic Name

The DataHub data domain name (in the example above the Access Name would be associated with the DataSim data domain).

Which protocol to use

Select DDE.

- Click OK to add the new Access name.

This Access Name will be associated with all InTouch tags that receive data from the DataHub.

Configuring InTouch Tag Names

To associate an InTouch tag with a DataHub point, the tag must be defined as an I/O type tag.

Read Only Tagnames

The image below shows a tagname configuration for a Read Only Write tagname. This means the value can only be read from the DataHub into your InTouch application.

The screenshot shows the 'Tagname Dictionary' dialog box with the 'Details' tab selected. The configuration is as follows:

- Tagname:** Sine
- Type:** I/O Real
- Group:** \$System
- Read only:** Selected (indicated by a filled circle)
- Read/Write:** Unselected (indicated by an empty circle)
- Comment:** (Empty text box)
- Log Data:** Unselected
- Log Events:** Unselected
- Retentive Value:** Unselected
- Retentive Parameters:** Unselected
- Initial Value:** 0
- Min EU:** -0.5
- Max EU:** 0.5
- Deadband:** 0
- Min Raw:** -0.5
- Max Raw:** 0.5
- Eng Units:** (Empty text box)
- Log Deadband:** 0
- Conversion:** Linear (Selected), Square Root (Unselected)
- Access Name:** DataHub
- Item:** Sine
- Use Tagname as Item Name:** Unselected

In the Tagname Dictionary window:

- Choose one of the I/O types of tagname.
- Click the AccessNames button and select the DataHub access name.
- In the Item field, enter the name of the corresponding point from the DataSim data domain in the DataHub.
- Enter values for Min EU, Max EU, Min EU, and Max Raw.
- Click Save, then click Close.

Read/Write Tagnames

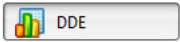
The image below shows a similar tagname configuration for a Read/Write tagname. This allows you to both read and write data back to the DataHub from your InTouch application.

The Tagname Dictionary dialog box is shown with the 'Details' tab selected. The 'Tagname' field contains 'Update' and the 'Type' is 'I/O Real'. The 'Group' is '\$System' and 'Read/Write' is selected. The 'Comment' field is empty. There are checkboxes for 'Log Data', 'Log Events', 'Retentive Value', and 'Retentive Parameters'. Below these are fields for 'Initial Value' (0), 'Deadband' (0), 'Eng Units' (empty), 'Min EU' (0), 'Min Raw' (0), 'Max EU' (20), and 'Max Raw' (20). There is a 'Log Deadband' field (0) and a 'Conversion' section with 'Linear' selected and 'Square Root' unselected. The 'Access Name' is 'DataHub' and the 'Item' is 'UpdateFrequency'. A 'Use Tagname as Item Name' checkbox is at the bottom right.

15.2. Getting data out of InTouch

The Cogent DataHub can act as a DDE client to your InTouch application. The following describes how to access InTouch data using a DDE Advise request from the DataHub to your InTouch application.

Configuring the Cogent DataHub

1. Right click on the Cogent DataHub system-tray icon and choose Properties.
2. In the Properties window, select DDE .

The DDE Client dialog box shows the 'Act as a DDE client to these services and topics:' checkbox checked. Below it is a table for 'Request Advise from these services/topics:' with columns 'DDE Connection Name' and 'Status'. To the right of the table are 'Add...', 'Edit...', and 'Remove' buttons.

3. Make sure the Act as DDE client box is checked.



For best performance, ensure that a DDE server (in this case, InTouch) 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.

4. Click the Add button. This opens the DDE Item Definition window where you can add InTouch as a new DDE service.

The DDE Item Definition dialog box shows 'Connection Name' as 'InTouch', 'Service' as 'VIEW', and 'Topic' as 'TAGNAME'. Below is a table for 'Item Names' with columns 'DDE Item', 'Point Name', and 'Data Domain'. The table contains one row: 'Tag_1', 'Tag_1', 'default'. To the right of the table are 'Add' and 'Remove' buttons. At the bottom are 'OK' and 'Cancel' buttons.

5. Enter the information in the DDE Item Definition window as follows:

Connection Name

Create a name for this link.

Service

VIEW

Topic

TAGNAME

Item Names

Enter the first InTouch tagname you want to read into the DataHub, and click the Add button. If you want to change the DataHub data domain into which the InTouch tagnames are read, click on the list of DDE Items and edit the Data Domain field. Continue adding names by entering them and clicking the Add button until all names have been added.

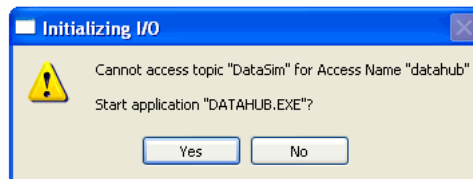
6. Once you have defined all the tagnames you want to read from your InTouch application, click the OK button.
7. Run your InTouch application and then view the DataHub Data Browser to see the data arrive from InTouch into the DataHub data domain you defined above.

15.3. Reading and writing data in both directions

Both the Cogent DataHub and InTouch are capable of working with DDE connections in both directions, so you can configure both as sources of data.

15.4. Error message displayed when starting InTouch WindowViewer

If you see this error message when you start your application in the WindowViewer program:

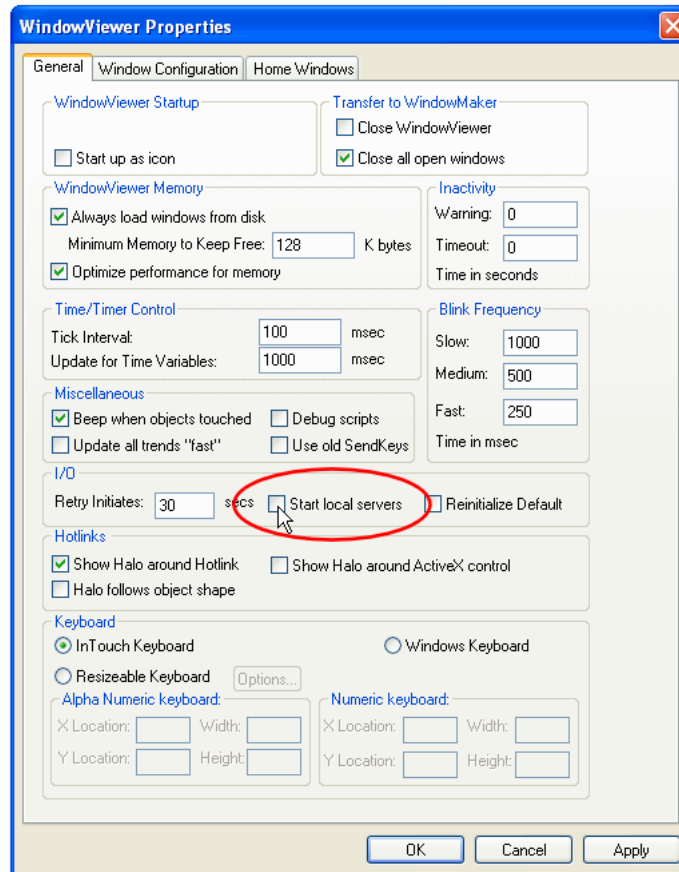


Then do the following:

1. Make sure the DataHub is running.
2. Click on the No button to ignore the error.

You can prevent this error message from happening again by changing the WindowViewer properties by following these steps:

1. In the WindowMaker program's Special menu, choose Configure, and then WindowViewer. The WindowViewer Properties window will be displayed:



2. Uncheck the **Start local servers** option in the **I/O** section of this window.
3. Click **OK**, then save your application and rerun the WindowsViewer application.

Chapter 16. DataHub Scripting

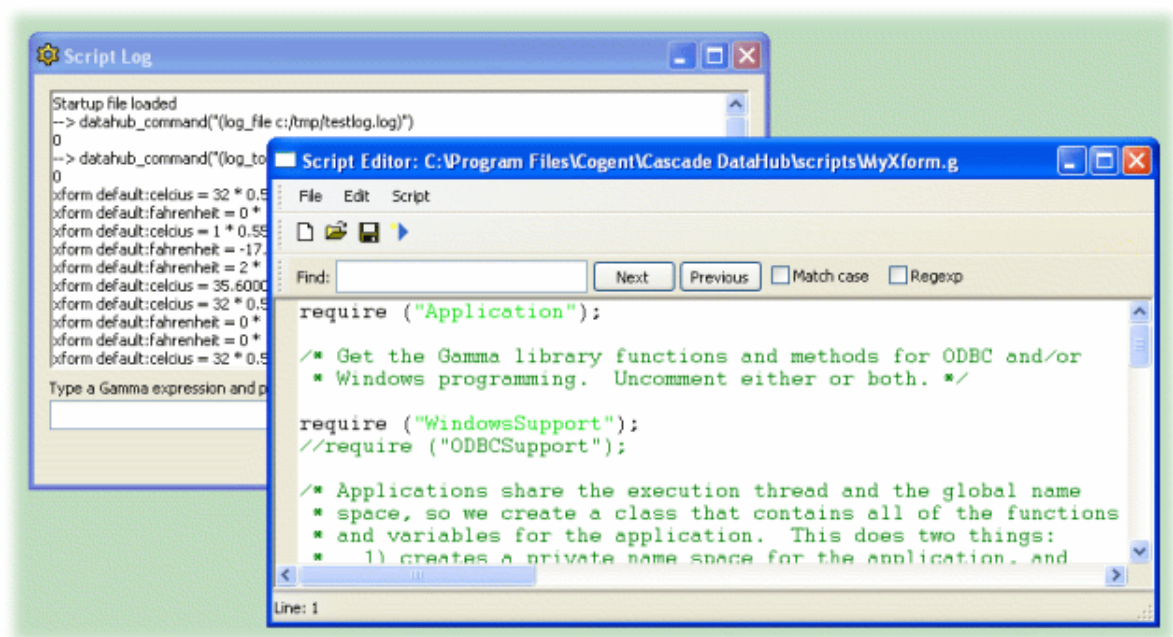
The DataHub has a powerful, built-in scripting language called *Gamma*. Using Gamma, you can write scripts to interact with the DataHub and its data in various ways, such as:

- 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.

16.1. Tools

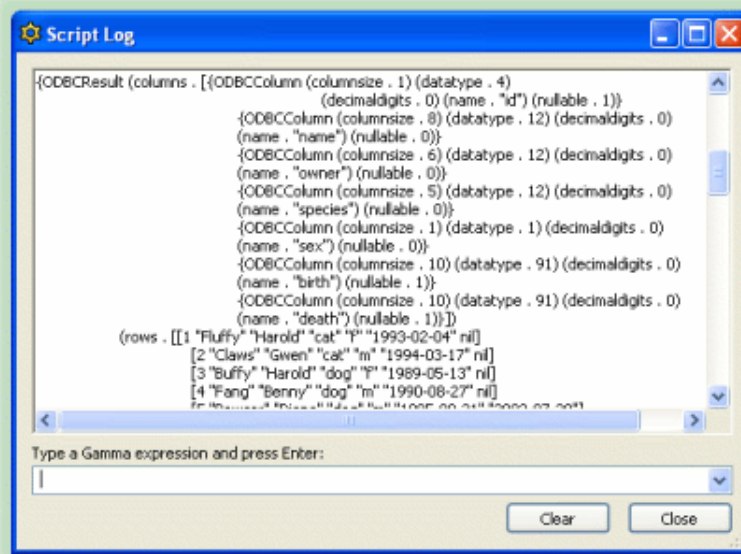
The DataHub comes with a built-in [Script Editor](#) for writing and editing scripts, as well as a [Script Log](#) for viewing script outputs.



Please refer to the DataHub Scripting manual for more information about how to use these tools.

16.2. DataHub ODBC (Open Database Connectivity) Scripting

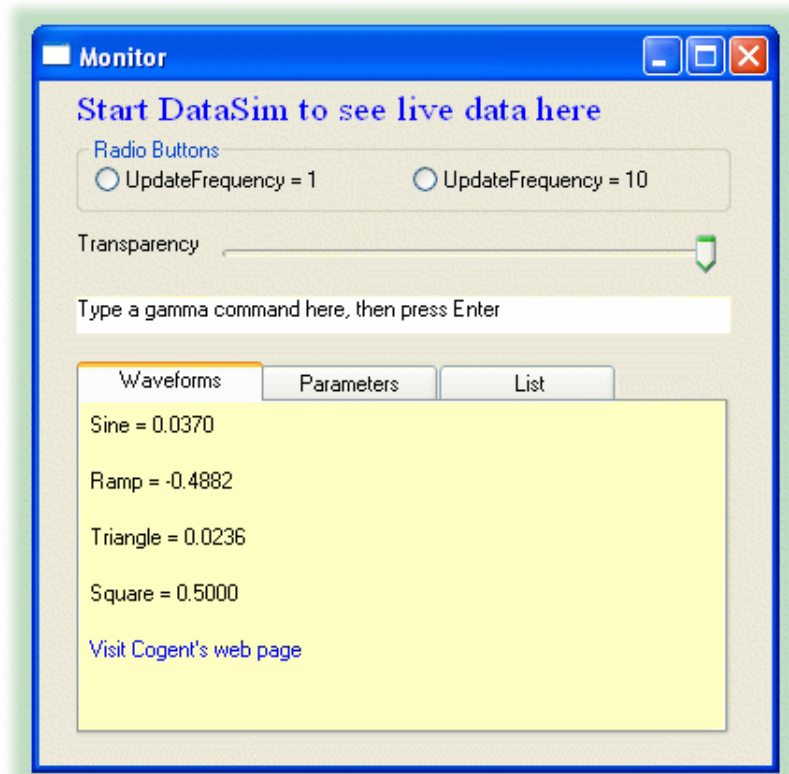
The ODBC support in the DataHub provides an interface to any ODBC-compliant database. It lets you create a class for any database table, and assign each column of the table as an instance variable of the class, giving you complete access to any point of data in the database.



Please refer to the DataHub ODBC Support manual for more information.

16.3. 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:



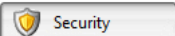
Please refer to the DataHub Windows Scripting manual for more information. The code for this example is in the `WindowsExample.g` file included in your distribution.

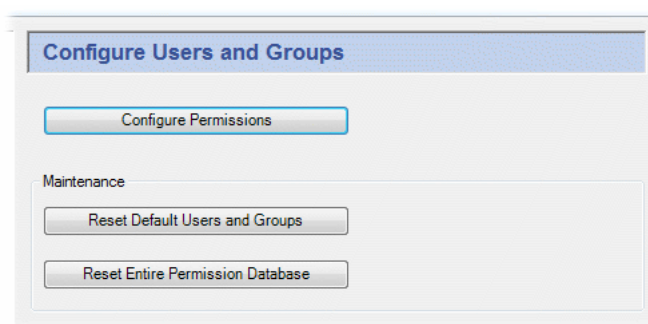
Chapter 17. Security

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.

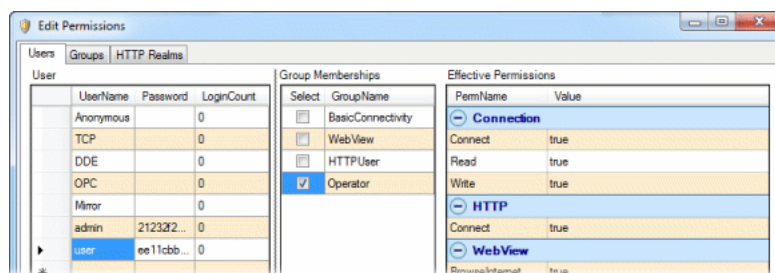
The DataHub also provides full [SSL \(Secure Sockets Layer\)](#) encryption for TCP/IP [tunnelling and mirroring](#) connections.

17.1. 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 three default groups: **Permissive**, **WebView**, and **HTTPUser**. 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**.

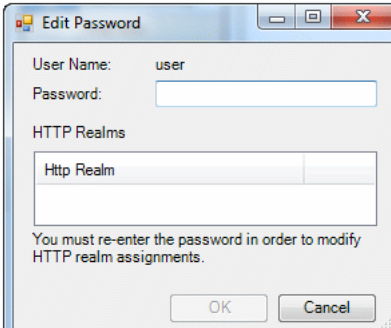
Users Groups HTTP Realms						
Group		Permissions				
	GroupName	Select	PermName	CombineType	Data Type	Value
	BasicConnectivity	Connection				
	WebView	<input checked="" type="checkbox"/>	Connect	AnyTrue	boolean	true
	HTTPUser	<input checked="" type="checkbox"/>	Read	AnyTrue	boolean	true
	Operator	<input checked="" type="checkbox"/>	Write	AnyTrue	boolean	true

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 [Section 17.3, User Authentication](#).

Users				Groups		HTTP Realms	
User				Group Memberships		Effective Permissions	
	UserName	Password	LoginCount	Select	GroupName	PermName	Value
	Anonymous		0	<input type="checkbox"/>	BasicConnectivity	⊖ Connection	
	TCP		0	<input type="checkbox"/>	WebView	Connect	true
	DDE		0	<input type="checkbox"/>	HTTPUser	Read	true
	OPC		0	<input checked="" type="checkbox"/>	Operator	Write	true
	Mirror		0			⊖ HTTP	
	admin	21232f2...	0			Connect	true
▶	user	ee11cbb...	0			⊖ WebView	

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:



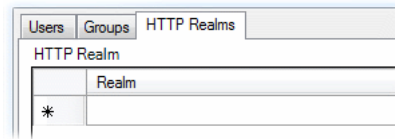
The dialog box is titled "Edit Password". It contains the following fields and controls:

- User Name:** A text field containing the value "user".
- Password:** A text field with a password mask (dots).
- HTTP Realms:** A list box containing the value "Http Realm".
- Message:** "You must re-enter the password in order to modify HTTP realm assignments."
- Buttons:** "OK" and "Cancel".

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 [Section 17.6, Passwords](#).

HTTP Realms

Here you can maintain a list of HTTP authentication realms. This list is accessed by the DataHub Web Server, as described here: [Section 20.9, 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 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.

17.2. SSL and Firewalls

The Cogent DataHub provides the option to use SSL encryption to protect your data when tunnelling/mirroring to another DataHub across a network connection. The SSL implementation uses the default SSL-3 encryption cipher: DHE-RSA-AES256-SHA, which is a 256-bit encryption method. The [Tunnel/Mirror](#) section of this manual explains how to configure SSL for tunnelling and mirroring.

SSL Certificates

An SSL certificate is required to use SSL encryption between DataHubs. The DataHub installs a default SSL Certificate for you, but you can use your own certificate if you prefer.

Because the DataHub is often used under circumstances where it is not possible or desirable to connect to the Internet, it does not check the issuing authority for a security certificate against the IP address or DNS name of the target computer, nor does it check the expiry date of the certificate. In most control applications it is not acceptable for the DataHub to refuse access to a critical process due to a simple mismatch in machine name declaration or the expiry of a time-limited certificate. Thus, any well-formed certificate will be accepted. Most important, the data encryption will be performed regardless of the validity of the certificate.

Firewall Ports

The DataHub lets you specify which ports it will use for tunnelling/mirroring over a network. Firewallled 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.

17.3. 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 not immediately disconnected, but instead keeps the permissions the were in force prior to the authentication attempt.

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 protocol does not provide a mechanism for authentication, this is the only mechanism available to limit the permissions of an OPC 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				Groups		HTTP Realms	
User	UserName	Password	LoginCount	Select	GroupName	PermName	Value
	Anonymous		0	<input type="checkbox"/>	BasicConnectivity	Connection	
	TCP		0	<input type="checkbox"/>	WebView	Connect	true
	DDE		0	<input type="checkbox"/>	HTTPUser	Read	true
	OPC		0	<input checked="" type="checkbox"/>	Operator	Write	true
	Mirror		0			HTTP	
	admin	21232f2...	0			Connect	true
	user	ee11cbb...	0			WebView	

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 only belong to one group at a time.

17.4. 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:

Connection

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.

Connection Read

This user is allowed to read point values and subscribe to point value changes.

Connection Write

This user is allowed to write a new point value to the Cogent DataHub.

Connection Force

If the user has Connection Write 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.

Connection CreatePoint

This user is allowed to create new points in existing data domains in the Cogent DataHub.

Connection DeletePoint

This user is allowed to delete a point from the Cogent DataHub.



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.

Connection 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.

Connection ChangeModel

This user can change the [organizational hierarchy](#) of the data points. This is the information that makes the data appear as a tree structure in the Data Browser.

Connection LoadConfig

This user is allowed to tell the Cogent DataHub to load a specific configuration file.

Connection Connection UserAdmin

This user is allowed to create and edit users and groups non-interactively. This permission is not currently available.

Connection Connection ConfigAdmin

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.

Connection Connection Shutdown

This user may transmit an **exit** command to the Cogent DataHub, causing it to shut down.

Restrictions**Restrictions LimitConcurrentLogins**

This user will be limited to the number of concurrent connections specified in **Restrictions MaxLoginLimit**, regardless of the connection type. For example, if the **Restrictions MaxLoginLimit** is 2, the user would be allowed to make two TCP connections, or one TCP and one DDE connection. This option also applies to anonymous users. This restriction is not currently being enforced.

Restrictions LimitTotalLogins

This user is allowed to connect to the Cogent DataHub at most the number of times specified by **Restrictions MaxLoginLimit**. Once the user has connected to the DataHub this many times, future attempts to log in will be refused, even after the DataHub has been restarted. This restriction is not currently being enforced.

Restrictions Expires

This user will be allowed to log in to the Cogent DataHub up to, but not including, the date specified by the **Restrictions ExpiryDate**. This restriction is not currently being enforced.

Restrictions MaxLoginLimit

An integer specifying the maximum total number of logins permitted, used by **Restrictions LimitTotalLogins**. This restriction is not currently being enforced.

Restrictions ConcurrentLoginLimit

An integer specifying the maximum number of concurrent logins, used by **Restrictions LimitConcurrentLogins**. This restriction is not currently being enforced.

Restrictions ExpiryDate

A date, used by **Restrictions Expires**. This restriction is not currently being enforced.

HTTP**HTTP Connect**

This user is allowed to connect to the DataHub Web Server.

WebView

WebView Connect

This user is allowed to connect to DataHub WebView. 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 WebView Connect permissions after the grace period expires, it will be disconnected.

WebView CreatePage

This user is allowed to create pages in DataHub WebView.

WebView CreateControl

This user is allowed to create controls in DataHub WebView. This function is not currently available.

WebView SavePage

This user is allowed to save his own pages in DataHub WebView.

WebView SaveControl

This user is allowed to save controls in DataHub WebView. This function is not currently available.

WebView DeletePage

This user is allowed to delete his own pages in DataHub WebView. This function is not currently available.

WebView DeleteControl

This user is allowed to delete controls in DataHub WebView. This function is not currently available.

WebView ViewPage

This user is allowed to view his own pages in DataHub WebView.

WebView EditPage

This user is allowed to edit his own pages in DataHub WebView.

WebView ViewOtherOwnerPage

This user is allowed to view pages belonging to other owners in DataHub WebView.

WebView EditOtherOwnerPage

This user is allowed to edit pages belonging to other owners in DataHub WebView.

WebView SaveOtherOwnerPage

This user is allowed to save pages belonging to other owners in DataHub WebView.

WebView DeleteOtherOwnerPage

This user is allowed to delete pages belonging to other owners in DataHub WebView. This function is not currently available.

WebView EditOtherOwnerControl

This user is allowed to edit controls belonging to other owners in DataHub WebView. This function is not currently available.

WebView SaveOtherOwnerControl

This user is allowed to save controls belonging to other owners in DataHub WebView. This function is not currently available.

WebView DeleteOtherOwnerControl

This user is allowed to delete controls belonging to other owners in DataHub WebView. This function is not currently available.

WebView ViewOnlineHelp

This user is allowed to view the DataHub WebView online help.

WebView BrowseInternet

This user is allowed to browse the Internet from within DataHub WebView.

WebView ChangeTheme

This user is allowed to change the DataHub WebView theme. This function is not currently available.

WebView ChangeOptions

This user is allowed to change DataHub WebView options.

WebView Troubleshoot

This user is allowed to use the troubleshooting abilities of DataHub WebView.

WebView ConfigureTraceSettings

This user is allowed to configure trace settings in DataHub WebView.

17.5. 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.

Command Name	Permissions Required
acksuccess	none
add	Connection Write
alive	none
append	Connection Write
assembly	Connection ConfigAdmin
attribute	Connection ConfigAdmin
auth	none
authgroup	Connection ConfigAdmin
authuser	Connection ConfigAdmin
auto_create_domains	Change auto domain creation
auto_timestamp	Connection ConfigAdmin
bandwidth_reduce	none
bridge	Connection ConfigAdmin, Connection Write
bridge_remove	Connection ConfigAdmin
bridge_transform	Connection ConfigAdmin
cforce	Connection Write, Connection Force
cread	Connection Read, Connection CreatePoint
create	Connection CreatePoint
create_domain	Connection CreateDomain
report	Connection Read, Connection CreatePoint
cset	Connection Write, Connection CreatePoint

Command Name	Permissions Required
cwrite	Connection Write, Connection CreatePoint
debug	Connection ConfigAdmin
defaultprop	Connection ConfigAdmin
delete	Connection DeletePoint
deleted	Connection DeletePoint
div	Connection Write
domain	none
drop_license	Connection Connect
dump	Connection ConfigAdmin
echo	Connection Write
enable_bridging	Connection ConfigAdmin
enable_connect_server	Connection ConfigAdmin
enable_dde_client	Connection ConfigAdmin
enable_dde_server	Connection ConfigAdmin
enable_mirror_master	Connection ConfigAdmin
enable_mirror_slave	Connection ConfigAdmin
enable_scripting	Connection ConfigAdmin
enable_tcp_server	Connection ConfigAdmin
error	none
exception_buffer	Connection ConfigAdmin
execute_plugin	Connection ConfigAdmin
exit	Connection Shutdown
failed_license	Connection ConfigAdmin
flush	Connection ConfigAdmin
force	Connection Write, Connection Force
format	Connection Connect
heartbeat	none
ignore	Connection Read
ignore_old_data	Connection ConfigAdmin
include	Connection LoadConfig
instance	Connection ConfigAdmin
load_config_files	Connection LoadConfig
load_plugin	Connection ConfigAdmin
load_scripts	Connection ConfigAdmin
lock	Connection Write
log_file	Connection ConfigAdmin
log_to_file	Connection ConfigAdmin
master_host	Connection ConfigAdmin
master_service	Connection ConfigAdmin
mirror_master	Connection ConfigAdmin
mirror_master_2	Connection ConfigAdmin
mult	Connection Write

Command Name	Permissions Required
on_change	Connection ConfigAdmin
point	Connection Write
private_attribute	Connection ConfigAdmin
property	Connection ConfigAdmin
quality	Connection Write
read	Connection Read
readid	Connection Read
register_datahub	Connection Read
report	Connection Read
report_all	Connection Read
report_domain	Connection Read
report_errors	Connection Read
request	Connection Read
request_initial_data	Connection Read
secure	Connection Write
set	Connection Write
show_data	Connection ConfigAdmin
show_debug_messages	Connection ConfigAdmin
show_event_log	Connection ConfigAdmin
show_icon	Connection ConfigAdmin
show_properties	Connection ConfigAdmin
show_script_log	Connection ConfigAdmin
slave	Connection Read
subassembly	Connection ConfigAdmin
success	none
sync	Connection Write
taskdied	Connection ConfigAdmin
taskstarted	Connection ConfigAdmin
tcp_service	Connection ConfigAdmin
timeout	none
transmit_insignificant	Connection ConfigAdmin
type	Connection ConfigAdmin
unload_plugin	Connection ConfigAdmin
unreport	Connection Read
version	none
warn_of_license_expiry	Connection ConfigAdmin
write	Connection Write
OPC-specific commands	
enable_opc_client	Connection ConfigAdmin
enable_opc_server	Connection ConfigAdmin
OPCAddItem	Connection Write
OPCAttach	Connection ConfigAdmin

Command Name	Permissions Required
OPCDetach	Connection ConfigAdmin
OPCInit	Connection ConfigAdmin
DDE-specific commands	
DDEAdvise	Connection Write
DDEConnect	Connection ConfigAdmin
DDEInit	Connection ConfigAdmin
DDEService	Connection ConfigAdmin
DDEUnadvise	Connection Write
DDEUnadvisePoint	Connection Write
EnableDDEServer	Connection ConfigAdmin

17.6. 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.

Chapter 18. Working With Data

This chapter gives an overview of how the Cogent DataHub handles data and the various protocols it works with.

18.1. 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.
- **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.

18.1.1. 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 startup. Sometimes referred to as *seeding*, this is done with supplemental configuration files. Please refer to [Section 1.6, Configuration Files](#) for more details.

18.1.2. 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.

18.1.3. Viewing Data Points

You can view the values of all data points with the [Data Browser](#).

18.1.4. 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.

18.2. Data Communication Concepts

These basic concepts of data communications will help you understand how the Cogent DataHub works.

18.2.1. 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*.

18.2.2. 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.

18.2.3. 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.

18.3. 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.

18.3.1. OPC Protocol

OPC (OLE for Process Control) is an interface specification for data communications that is popular in industrial environments. Please refer to [Appendix E, OPC Overview](#) for general information about OPC.

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](#). If acting as a client, it will automatically attempt to find or start the OPC server that has been configured, and then start receiving data. If acting as a server, it will automatically respond to requests from any OPC client on the system.

OPC Items and Properties

The OPC protocol uses the concept of an *item* as a way to structure data. Every item has 6 required *properties*: Value, Timestamp, Quality, Access Rights, Scan Rate, and Canonical Type. As users of the data, we are mostly interested in Value, Timestamp, and Quality. 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:

Property	Current value
Value	47.2
Timestamp	Apr 27 16:27:24.300
Quality	Good
High	60.0
Low	38.5
Alarm Level	54.0
Engineering Units	Celsius
Description	Temperature of Tank A

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 Cogent 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 Cogent 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, any filter you run apply to those items as well.

18.3.2. 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 communication with Windows programs such as Excel. Which of these commands you use depends on how you plan to control the flow of data between the spreadsheet and the Cogent 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 [Section 6.2.2, 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 [Section 6.1.2, 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 terminated.

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 [Section 6.1.1, 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 [Section 6.2.1, Method 1 - Configuring DDEAdvise loops in the Cogent DataHub](#).

For more information on DDE, please refer to [Appendix F, DDE Overview](#).

18.3.3. TCP and Tunnelling/Mirroring

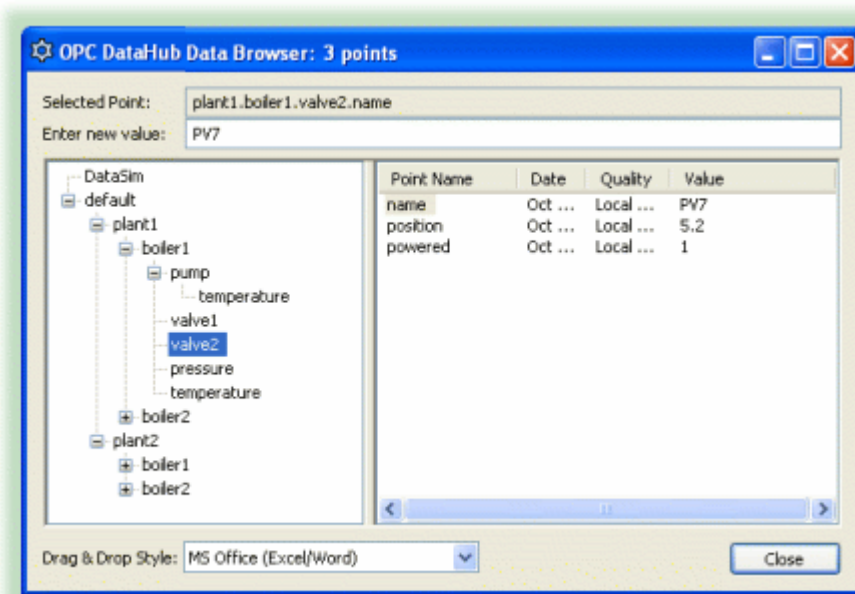
The Cogent DataHub uses TCP to communicate over a LAN, WAN, or the Internet. You can join two or more copies of the Cogent DataHub together over TCP and share exact copies of the data through data *tunnelling/mirroring*. Tunnelling/Mirroring means that the data and any updates to that data on one DataHub are exactly tunneled/mirrored across the network onto any other DataHub that is connected. Once a tunnelling/mirroring connection is established each participant maintains and updates an identical data set, as simultaneously as the TCP connection will permit. For more information on tunnelling/mirroring, please refer to [Section 20.4, Tunnel/Mirror](#).

18.3.4. The DataHub API

The DataHub APIs for C++, Java, and .NET lets any TCP-enabled program interface with the DataHub.

18.4. 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.



18.4.1. Data Domains

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.

18.4.2. 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...
      Attribute...
        Property...
        Property...
        Property...
    Subassembly
      Subassembly
        Attribute...
        Property...
        Property...
      Attribute...
      Attribute...
        Property...
  Assembly...

```

and so on.

The written syntax for all of these levels uses a dot (.) to divide the names, rather than a colon that was used for the data domain name. Hence, the syntax of point in a property in an attribute in a subassembly in an assembly in a data domain would be:

```
domain:assembly.subassembly.attribute.property
```

Properties 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.

18.4.3. Attributes and Types

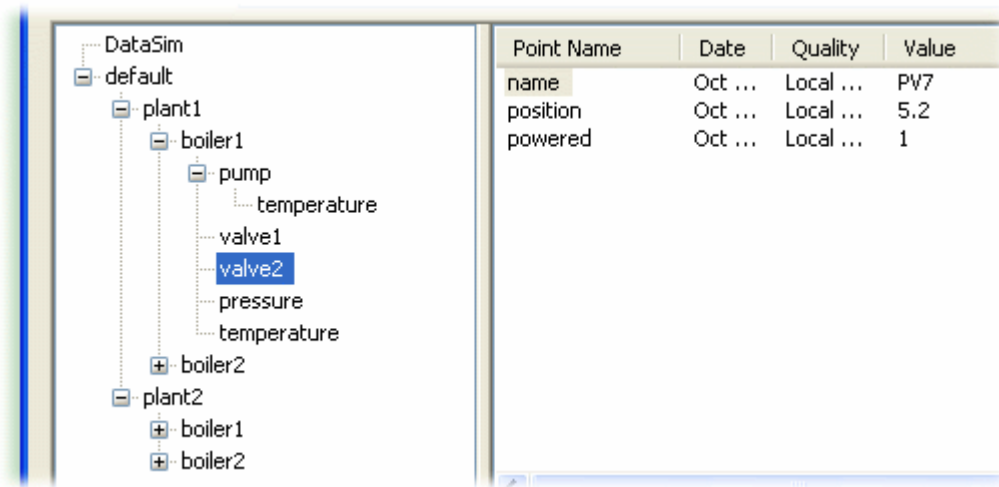
It is common for attributes to contain the same type of information. For example, all temperatures in a system are likely to share units, high alarm level, and value. To avoid repeating this information for each and every temperature in the system, we use a *type*. A type is the prototype, or class, 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](#).)

18.4.4. 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:



You can have the DataHub load this `plant.cfg` configuration file on startup (see [Section 1.6, Configuration Files](#)).

```

;;; 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 that can 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 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 Powervalue Valve)
(property default Powervalue 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 Powervalue valve2)

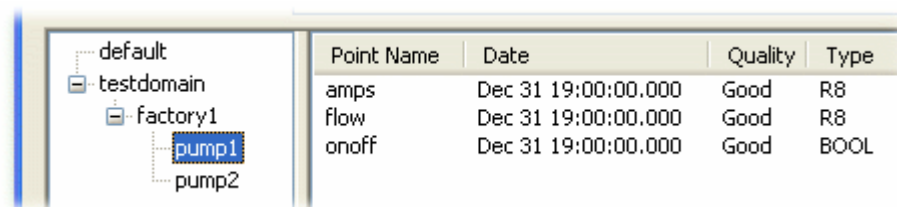
;;; Create two plants named plant1 and plant2. These actually
;;; create the data points in the DataHub and arrange them in
;;; the hierarchy specified above. Up to this point, the commands
;;; have just been building a model. These calls instantiate the
;;; model.

(instance default plant1 Plant)
(instance default plant2 Plant)

```

18.4.5. 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)
```

Chapter 19. 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.

19.1. 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, [Procedure 19.2, 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, [Section 19.7.2, DataHub C++ API](#) for details.
- Always use binary mode when writing programs in Linux or QNX with the Cogent C API. Please refer to Optimizing Throughput in the Cogent C API manual for details.

19.2. Tunnel/Mirror (TCP) Connections for Slow Networks

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 timeout value is usually at least twice the heartbeat rate, and the default setting is five times the heartbeat rate. However, if you have a very slow or irregular network connection, the best thing to ensure that the connection remains open is to override the DataHub timeout, and use the TCP timeout mechanism. It takes longer to detect a network failure, because it waits as long as the TCP stack waits. It means the slowest recognition of a broken link, but doesn't give up until TCP says the link is really dead.

How to Optimize

- To ensure the longest-lasting TCP connection over a very slow network, you need to set the heartbeat Timeout to 0. This will cause the DataHub to ignore its heartbeat settings and rely on the TCP implementation for detecting a broken connection. Please refer to How to Optimize, [Procedure 19.3, Slow network](#) for details.

19.3. 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 [Section 19.6, 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, [Procedure 19.4, 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, [Section 19.7.3, Gamma scripts](#) for details.

19.4. 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, [Procedure 19.4, Old value queueing and un-buffered delivery](#) for details.

19.5. 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.

19.6. 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.

19.7. How to Optimize

19.7.1. Tunnel/Mirror (TCP) connections

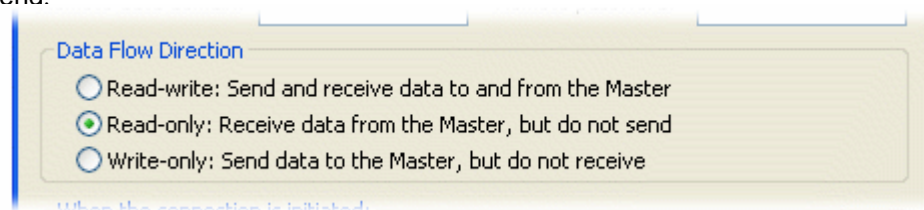
Read-only connections

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.

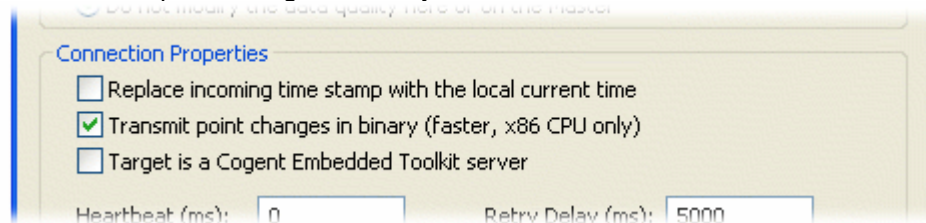


4. Click Apply.

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.

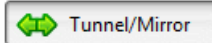


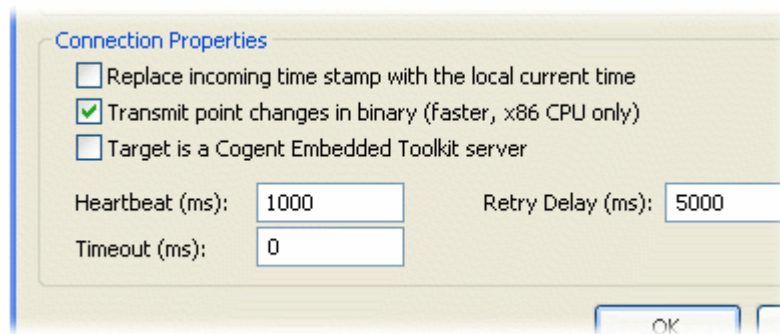
5. Click Apply.

For more information, please refer to [Section 19.1, Binary Mode Tunnel/Mirror \(TCP\) Connections](#).

Slow network

To optimize performance for a slow network, you can set the heartbeat timeout to 0 on the **slave** side, 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 Connection Properties section, enter 0 in the Timeout entry field.



4. Click Apply.

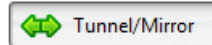
For more information, please refer to [Section 19.2, Tunnel/Mirror \(TCP\) Connections for Slow Networks](#).

Old value queueing and un-buffered delivery

Configure the **master** side of the tunnel/mirror connection:

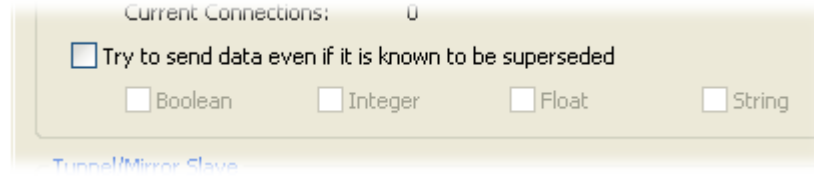
1. On the DataHub that is the *master* for the tunnel/mirror connection, right click on the DataHub system-tray icon and choose Properties.

- In the Properties window, select Tunnel/Mirror

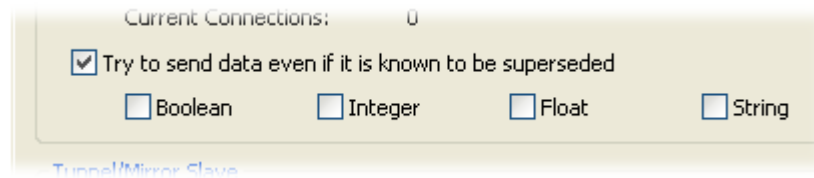


- In the Tunnel/Mirror Master section, you can choose between one of three states:

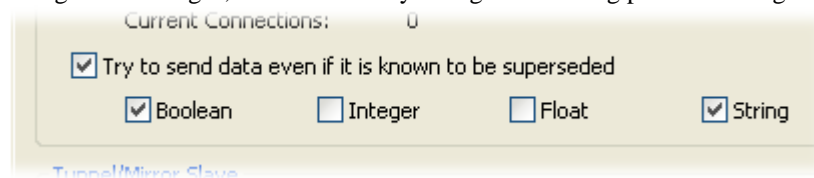
- **No queuing with buffered delivery.** This is the fastest state. To do this, un-check the option Try to send data even if it is known to be superseded in the Tunnel/Mirror configuration tab:



- **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 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:



- Click Apply.

For more information, please refer to [Section 19.3, Old Value Queuing](#) and [Section 19.4, Un-Buffered Delivery](#).

19.7.2. DataHub C++ API

Binary mode connections

- In your program, call the method

```
CDataHubConnector::sendBinaryPointMessage(bool enable)
```

to enable or disable binary messages.

For more information, please refer to [Section 19.1, Binary Mode Tunnel/Mirror \(TCP\) Connections](#).

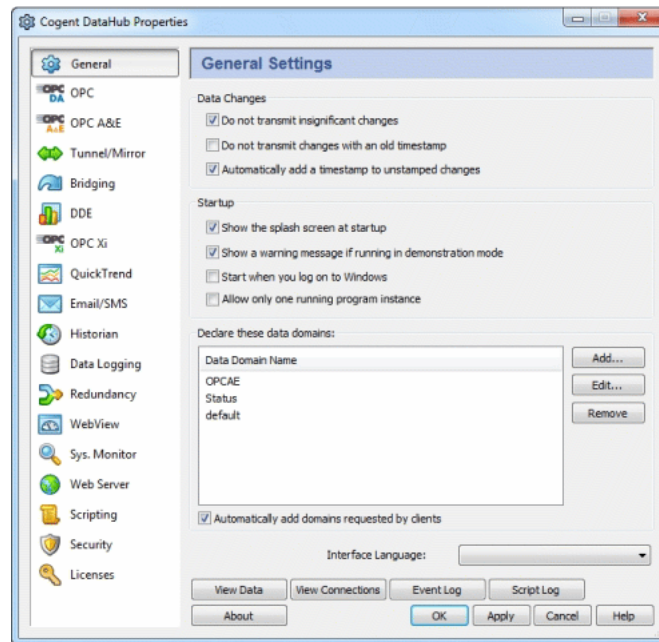
19.7.3. Gamma scripts

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.

Chapter 20. Properties Window

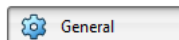
This is where you can configure the Cogent DataHub.



For all the options (General, Data Domains, DDE, etc.) in this window:

- 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.

20.1. General



This first option in the Properties Window lets you control how changes to data are transmitted from the Cogent DataHub.

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 19:00:00.000). If this button is *unchecked* and the **Do not transmit** change with an old timestamp is *checked*, then any changes with a 0 timestamp won't get transmitted at all.

Startup

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

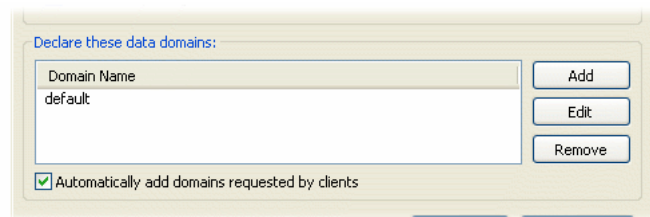
Show a warning message if running in demonstration mode lets you hide or show the message telling you the demo will terminate in one hour.

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

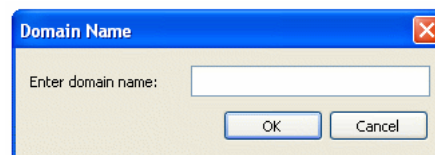
Allow only one running program instance prevents more than one Cogent DataHub from running at the same time.

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 [Section 18.4.1, 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.

Other Options

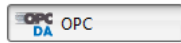
The **Interface Language** list lets you choose a language for the interface. If you don't find your language, you can contact Cogent for instructions on how to add a translation to the DataHub source code.

The **View Data** button starts [Data Browser](#).

The **Event Log** button starts [Event Log](#).

The **Script Log** button starts [Script Log](#).

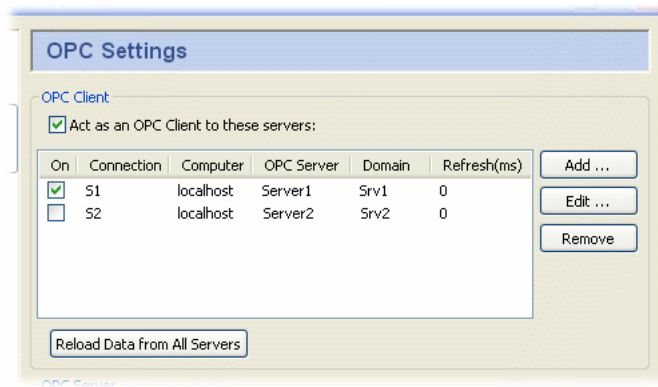
20.2. OPC DA



The **OPC option** lets you configure the Cogent DataHub to act as an OPC DA (Data Access) server, an OPC DA client, or both simultaneously. For more information on OPC, please refer to [Section 18.3.1, OPC Protocol](#) and [Appendix E, OPC Overview](#).

OPC Client

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



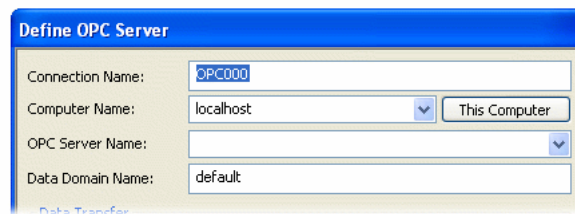
Check the **Act as an OPC Client** box for OPC client functionality. Since the DataHub can be a client to more than one OPC server, you need to specify server information for each OPC client connection. Once you have a server listed, you can activate or deactivate the connection using its **On** check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

The Define OPC Server Window

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



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 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 available 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 support consists of browsing support and support for the WriteVQT (Value, Quality, Timestamp) methods of the DA 3.0 specification. Normally, the Cogent 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 Cogent 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 Cogent 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-nnnnnnnnnnnnn} where each *n* is a hexadecimal digit. In this case, the Cogent 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 registry value:

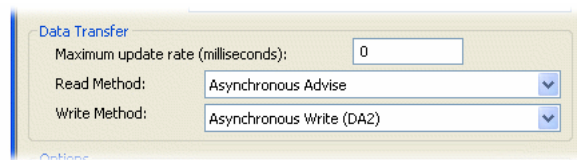
HKEY_CURRENT_USER\Software\Cogent\OPC DataHub\BrowseDA3

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

Options

- ☐ Treat OPC item properties as DataHub points where possible
- ☐ Read-only: Mark all items as Read-Only and disable writes to this server
- ☐ Only transmit GOOD quality data to this server
- ☐ Replace item time stamps with local clock time
- ☐ Force connection to use OPC DA 3.0
- ☐ Never use OPC DA 3.0
- ☒ Set failed incoming values to zero
- ☐ Never use OPC DA 2.0 BROWSE_TO function
- ☐ Never attach to an in-process COM server

Wait for server running state: 5000 ms

Pause before reading data: 1000 ms

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 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 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

This setting will remove 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.

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 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

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 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 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.

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

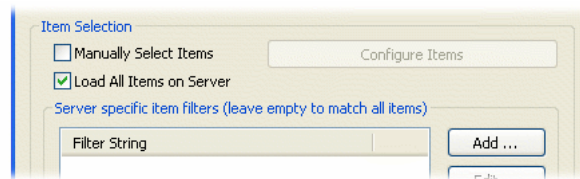
This parameter specifies a time for the DataHub to pause before reading the OPC server's data set. Some OPC servers report that they are running, but have not yet received the full data set from the process. If the DataHub attempts to connect right away, it might get a partial data set. The pause is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



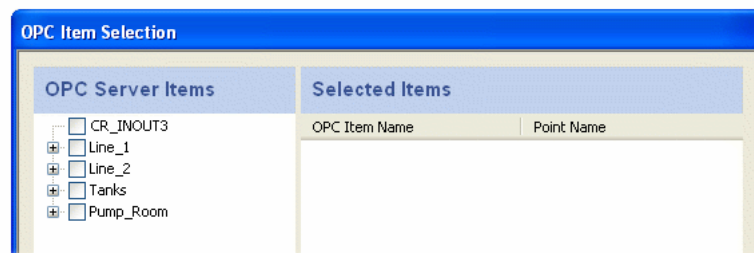
Manually Select Items



[Click here to watch a video.](#)



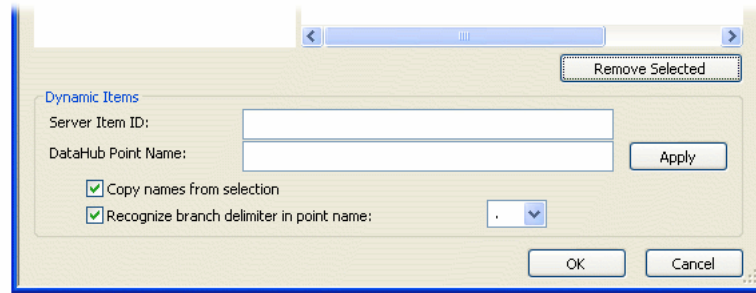
Check the **Manually Select Items** box and press the **Configure Items** button to open the OPC Item Selection window, where you can specify exactly which points you wish to use:



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.

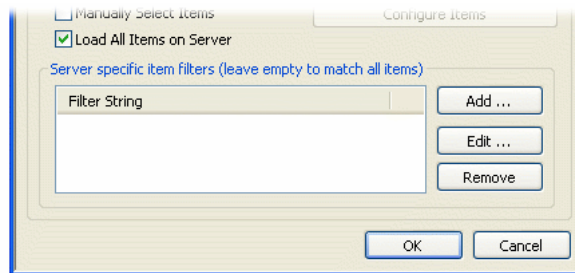
- To highlight a list of consecutive server items, click the first item, hold down the **Shift** key, and then click the last item. To highlight separate server items, hold down the **Ctrl** key as you click each item. To select a group of highlighted items, use the **Spacebar**.
- Selecting a server 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.



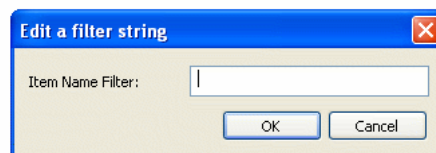
You may also configure dynamic items on the server. As you type in the **Server Item ID**, the system will fill in an identical **DataHub Point Name** for you (which you can change at any time). Press the **Enter** key or the **Apply** button to create the item. Checking the **Copy names from selection** 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** 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:

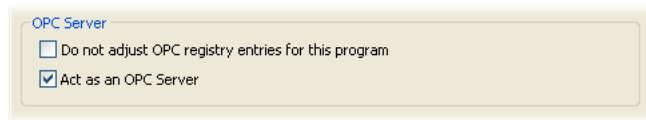


Enter a string that matches an item name, or a pattern to match multiple names. Each OPC server has its own syntax for pattern matching, so you may have to experiment a little to get exactly the points you need. Commonly, the symbol ***** matches any number of characters, while the symbol **?**

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 Server

The Cogent DataHub can act as a server to any number of OPC clients.



Check the Act as an OPC Server box to have the Cogent DataHub function as an OPC server.



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

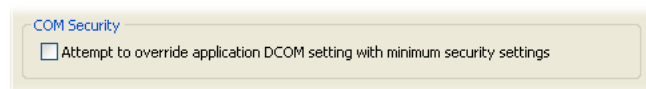
The Do not adjust OPC registry entries for this program option tells the Cogent DataHub not to alter its registry settings. This is useful if you want to use the Cogent DataHub with a redundancy server or some other program that modifies the DataHub's registry independently. Without this box checked, 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 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.

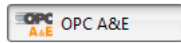
COM Security

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 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.

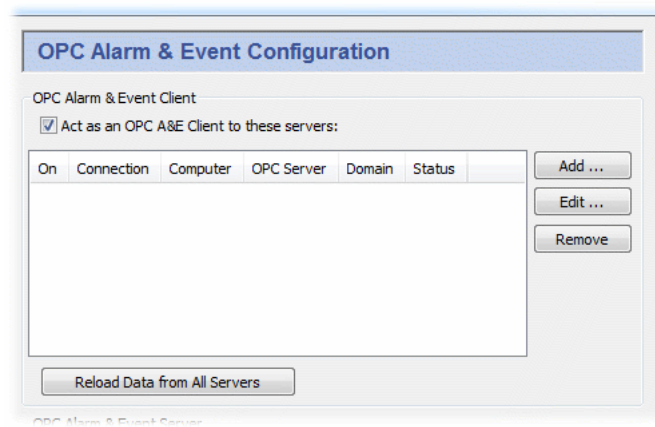
20.3. OPC A&E



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.

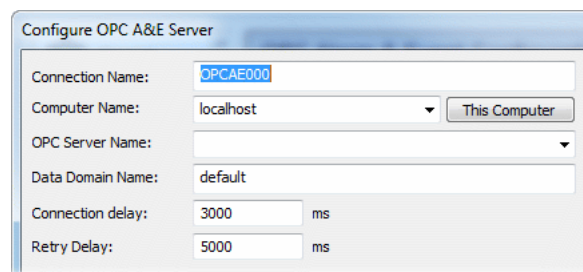


Check the Act as an OPC A&E Client box for OPC A&E client functionality. Since the DataHub can be a client to more than one OPC A&E server, you need to specify server information for each OPC A&E client connection. Once you have a server listed, you can activate or deactivate the connection using its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

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:



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

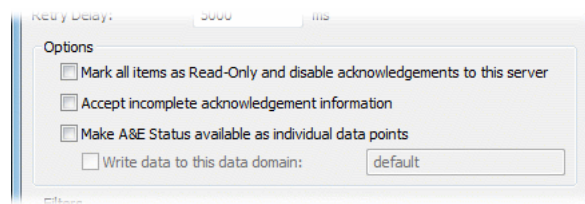
The number of milliseconds to delay the initial connection.

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

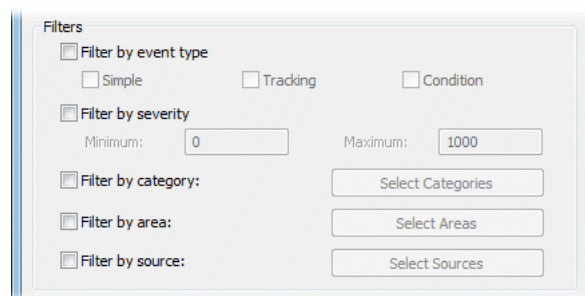
This allows connections to OPC A&E clients which are not configured for acknowledgements or that don't support this part of the OPC A&E specification.

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:

**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.

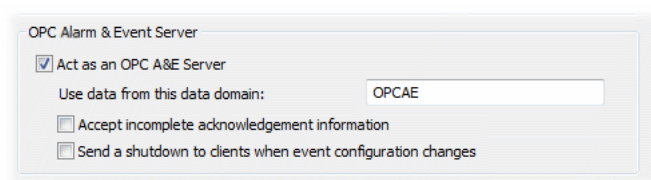
Filter by source:

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.

20.4. 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 OPC or DDE data across a network robustly and securely. Tunnelling is done 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. For example, the Cogent DataHub can tunnel (mirror) data to the Cascade DataHub running in the Linux or QNX operating systems. Please refer to the Mirroring Data section of the Cascade DataHub for Linux and QNX manual for details about those operating systems.

Direct TCP connections

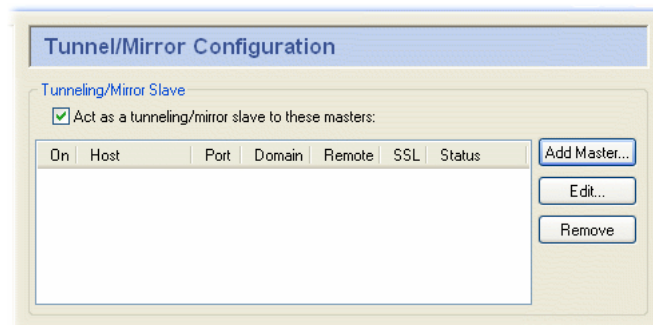
In addition to tunnelling, the Cogent DataHub can accept direct connections from any TCP client using the DataHub APIs for C++, Java, and .NET, such as DataSim or other, custom applications.

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.

Tunnelling/Mirroring 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:

Tunnel/Mirror Master Configuration

Primary Host: Port: 4502 ☐ Secure (SSL)

Secondary Host: Port: 4502

Local data domain: Remote user name:

Remote data domain: Remote password:

Data Flow Direction

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.

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 above.)

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.



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

- **Primary Host:** `developers.cogentrts.com`
- **Port:** 4502
- **Local data domain:** `test`
- **Remote data domain:** `test`

You now have 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.

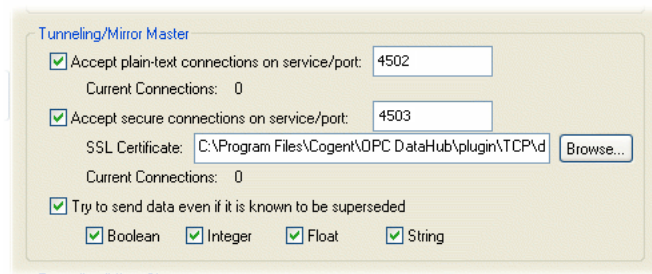
4. **Connection Properties** gives you these options:
 - **Replace incoming timestamp...** lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.
 - **Transmit point changes in binary** gives 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 a Cogent 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 [Section 19.2, Tunnel/Mirror \(TCP\) Connections for Slow Networks](#) for details.
- **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

Tunnelling/Mirroring Master

You can configure your DataHub to act as a master for either plain-text tunnelling, secure tunnelling using SSL, or both. Each mode uses a separate port number or service name.



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.

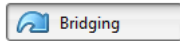
The DataHub installs an **SSL Certificate** for you. If you wish to move it or use a different one, you can change the directory path here. The SSL implementation uses the default SSL-3 encryption cipher: DHE-RSA-AES256-SHA. This is a 256-bit encryption. The server and client negotiate the best encryption 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.

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 [Section 19.7, How to Optimize](#).

20.5. Bridging

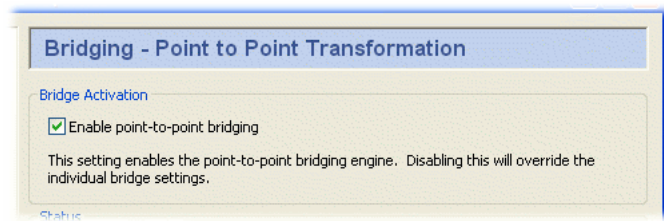


The **Bridging option** lets you configure the Cogent DataHub to configure data bridging. Bridging means connecting points from two different DataHub clients so that when one point changes, its value gets written to the bridged point.

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

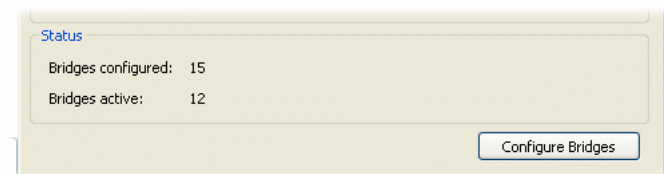
This setting enables and disables bridging globally for the whole Cogent DataHub program.



Ensure the box is checked to enable bridging. Uncheck the box to disable bridging.

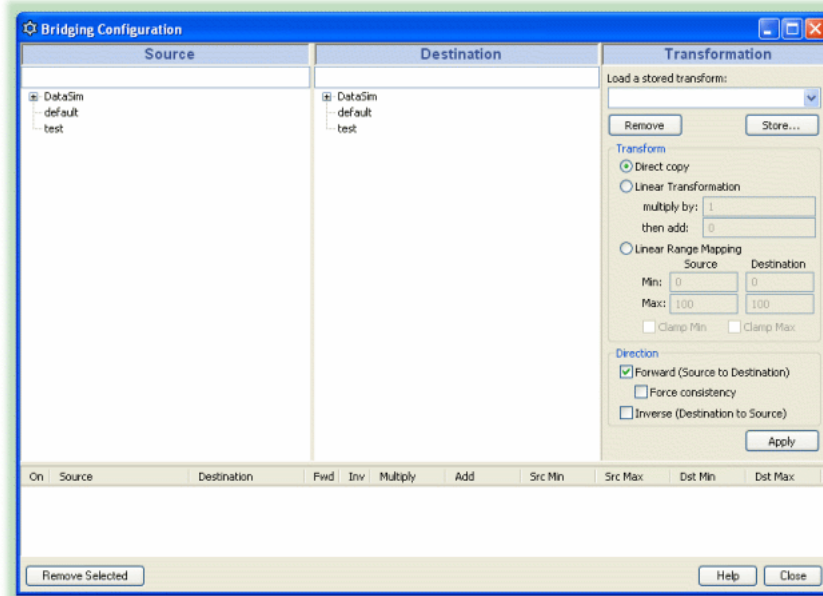
Status

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



Configure Bridges

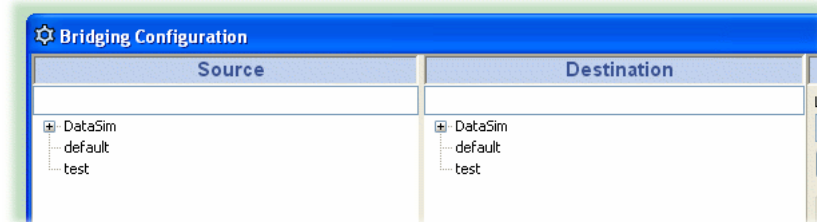
Click the Configure Bridges button to open the Bridge Configuration window:



The following sections give an overview of bridge configuration. For step-by-step instructions, please refer to [Section 5.2, Configuring Bridges](#).

Point Selection

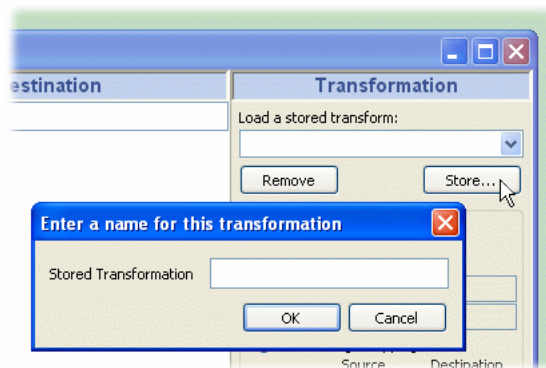
This is where you select the points to be bridged—a source point and a destination point.



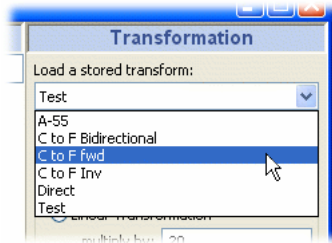
You can click on the point you need, or enter the name in the data-entry box at the top of the column.

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 Transformation**.



- **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**.

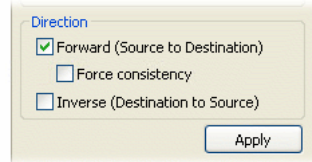
- **Linear Range Mapping** lets you enter a range for the source and destination, and the DataHub automatically calculates the corresponding linear transformation. For example, to create the same Fahrenheit to Celsius transformation, you could use the defaults of 0 and 100 for the **Min** and **Max** 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, ie. 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.



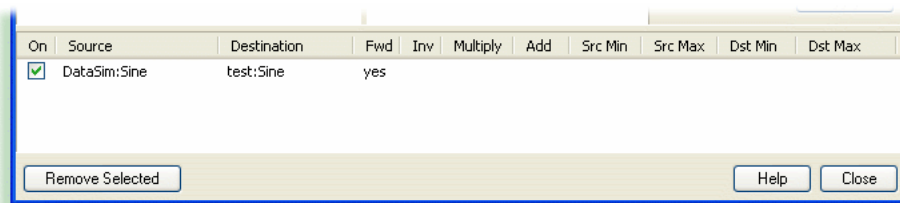
Selecting **Inverse** will apply the inverse of the transformation, as explained above.

- 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.

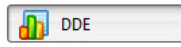
Point Display

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.

20.6. DDE



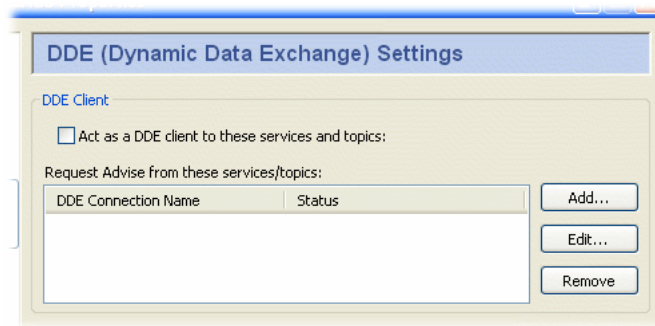
The [DDE option](#) lets you configure the Cogent DataHub to act as a DDE client and/or DDE server for **DDEAdvise** messages. For more information on DDE, please refer to [Section 18.3.2, DDE Protocol](#) and [Appendix F, DDE Overview](#).

DDE Client

To act as a DDE client, check the Act as a DDE client box.

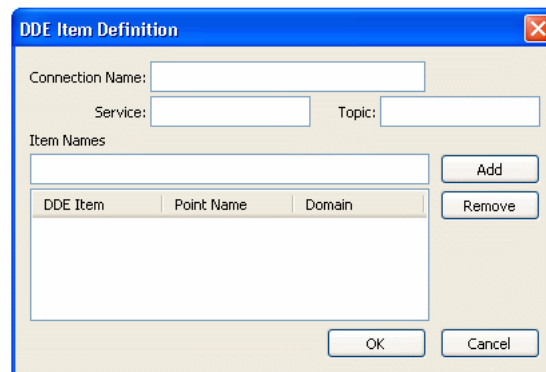


For best performance, ensure that a DDE server is running when using the DataHub as a DDE client. A DDE client can consume substantial system resources trying to connect if a DDE server is not available.



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.

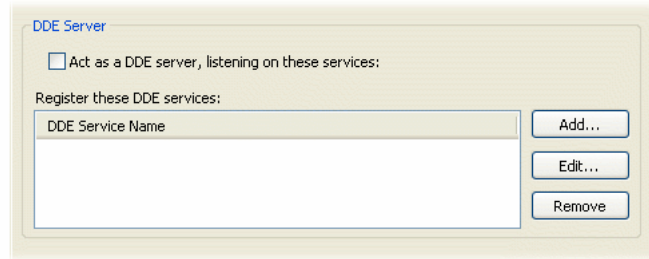
Topic

The DDE topic under which the data will be sent.

Item Names

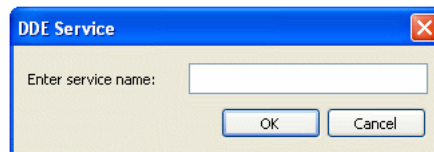
The DDE item name, which corresponds to a point name in the Cogent DataHub. If that point doesn't exist in the DataHub, it will be created. Enter a name for each item and press the **Add** button to add items. To remove an item, highlight it and press the **Remove** button. Once all the information is 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.



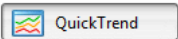
It is currently possible to have more than one instance of the Cogent DataHub, with one or more DDE service names in common, running on a single machine. If you plan to configure a system like this, you should ensure that each instance of the DataHub has unique data domain names. Otherwise, when any 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

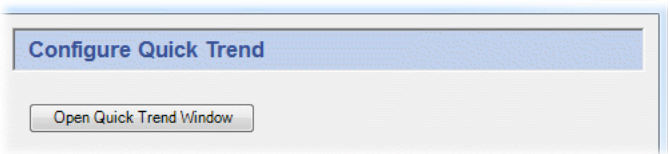


Checking this box will cause Excel to send strings of Unicode characters correctly, although slower than numerical data.

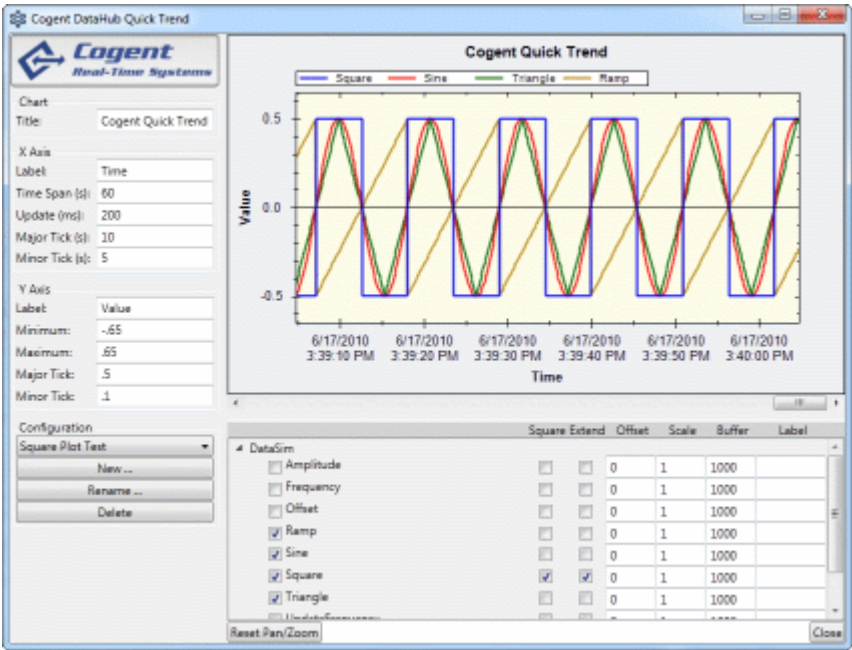
20.7. Quick Trend



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.

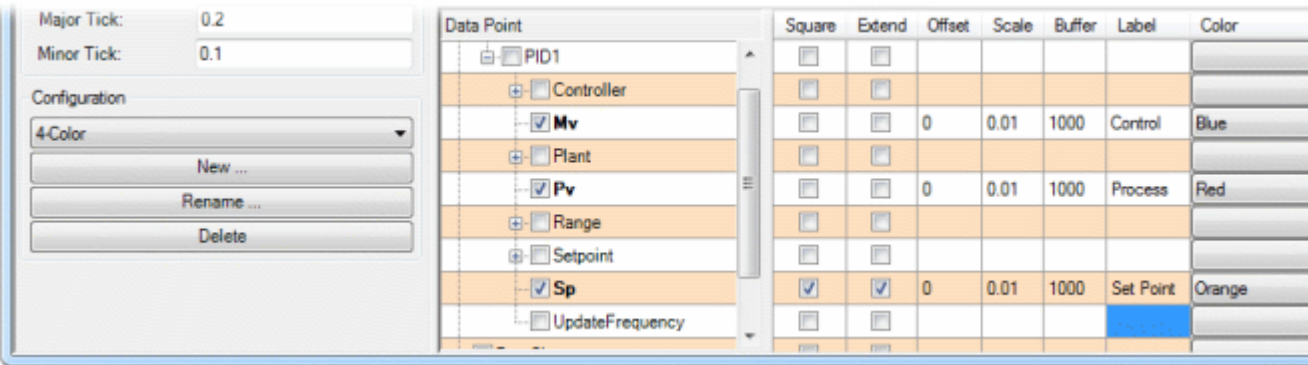


To configure a quick trend, press the Open quick trend window button:



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.

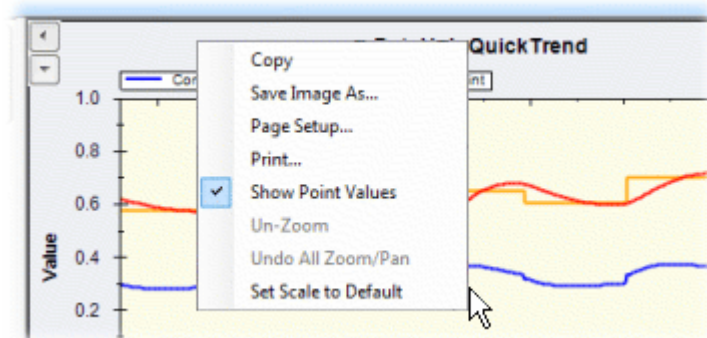
Label

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.
- For other features, the QuickTrend display has a menu available on a right mouse-click.

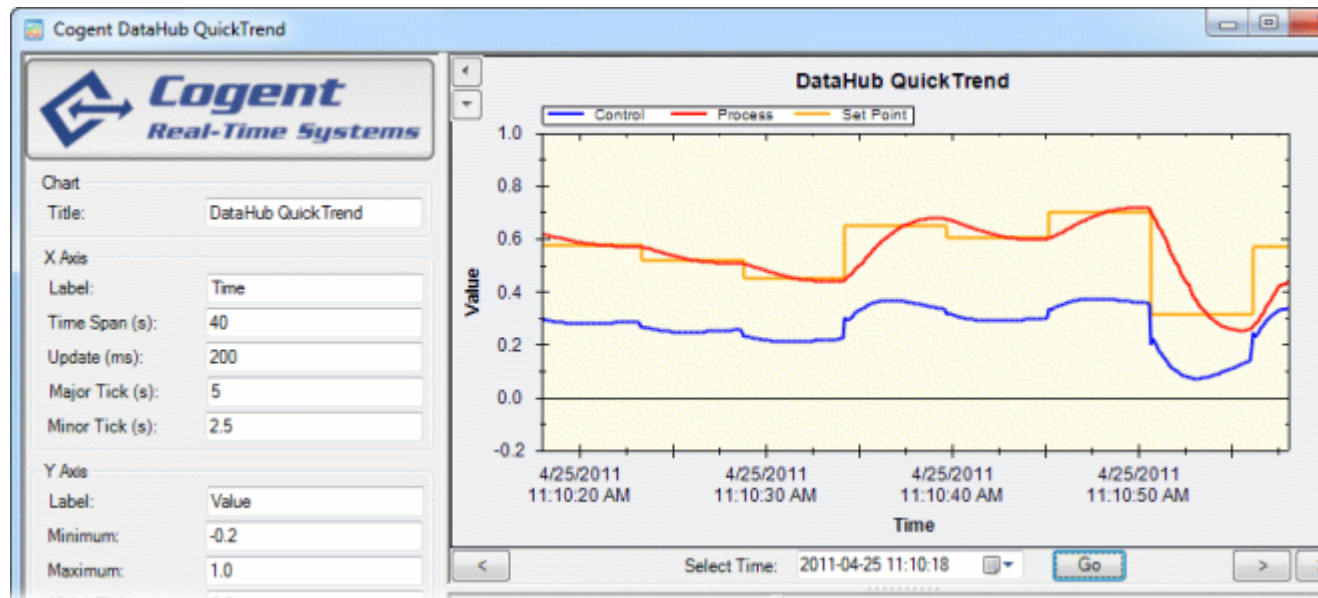


Using this menu you can copy, save, or print the current display, as well as unzoom and undo a zoom or pan. You can also hide or show point values, and reset the scale to the default.

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.

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.

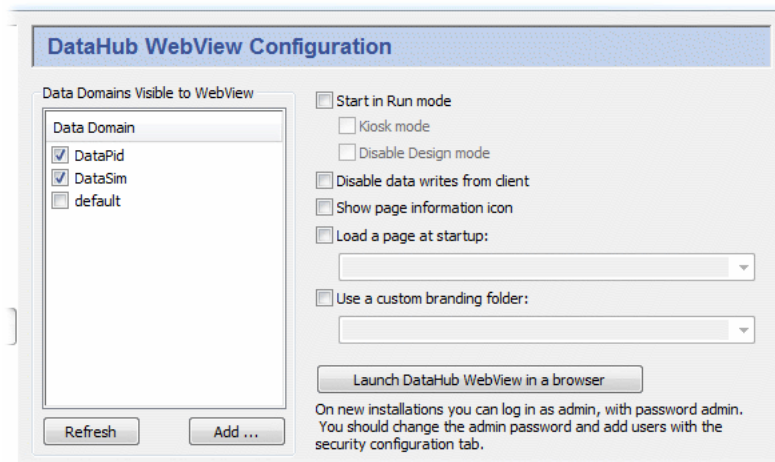
20.8. DataHub WebView



The [DataHub WebView](#) option is a web-based data visualization tool that provides a browser-based editor for designing animated displays of DataHub data that can be viewed using a standard web browser from anywhere on the Internet or corporate network.



This section covers configuration parameters for DataHub WebView. For information about 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

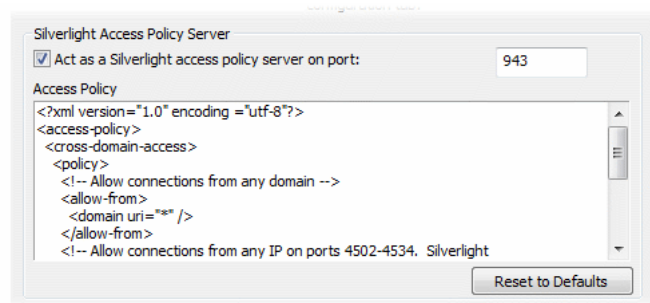
Allows you to specify a folder for holding custom branding information. For details, please refer to in the DataHub WebView manual.

Launch DataHub WebView in a browser

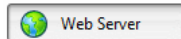
Provides a convenient way to start DataHub WebView to check this configuration.

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.



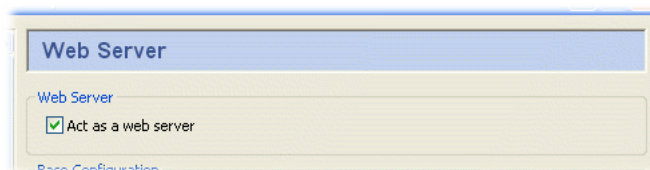
20.9. Web Server



The [Web Server option](#) lets you configure the 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](#).

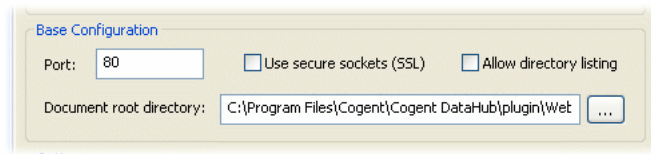
Web Server

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 [Section 8.2, Configuring the Web Server](#) for more details.

Use secure sockets (SSL)

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

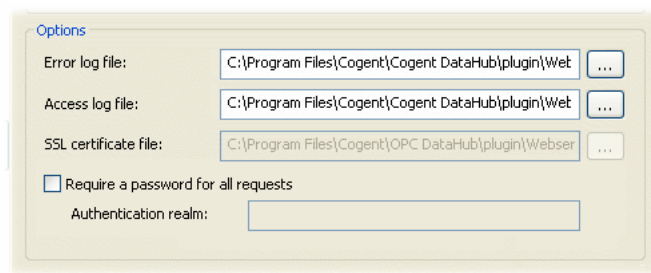
Allow directory listing

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:

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).

Require a password for all requests

Applies password security as configured in [Security](#).

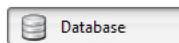


The security model changed from version 7.1 to version 7.2. User names and passwords in the Security tab will be maintained when moving from V7.1 to V7.2. User names and passwords in the Web Server tab will be lost. When upgrading to V7.2 you must re-assign Web Server realms to any relevant users in 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.

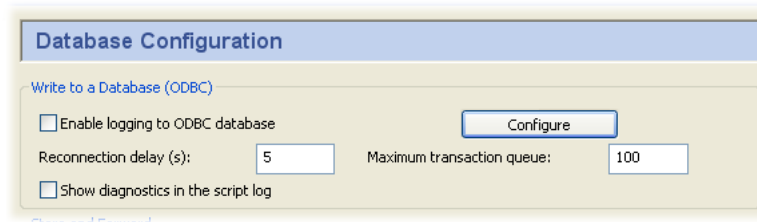
Authentication realm

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

20.10. Database



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



Write to a Database (ODBC)

Enable logging to ODBC database

Globally enables or disables all ODBC logging activity.

Reconnection delay (s):

Specifies the number of seconds before a reconnect is attempted if the ODBC connection is broken.

Maximum transaction queue

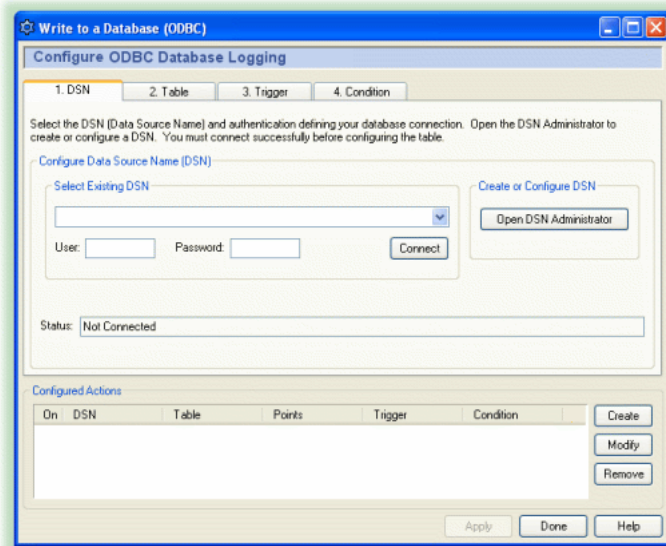
The DataHub maintains an in-memory queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. You can modify the depth of this queue to reduce the chance of involving the disk during busy periods. The queue depth for logging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

Show diagnostics in the Script Log

Puts diagnostic messages about the connection in the [Script Log](#).

Configure button

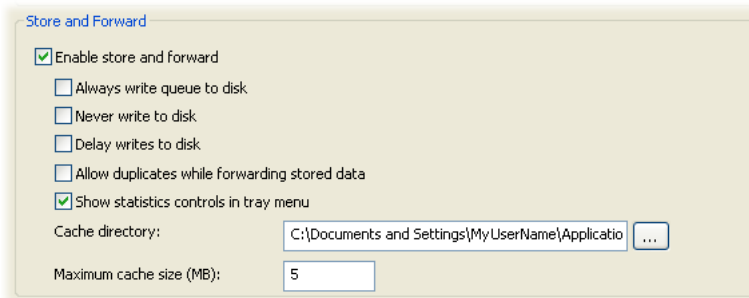
The Configure button opens the ODBC Data Logging Window.



For detailed instructions on using this interface, please refer to [Chapter 9, 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 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.



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).

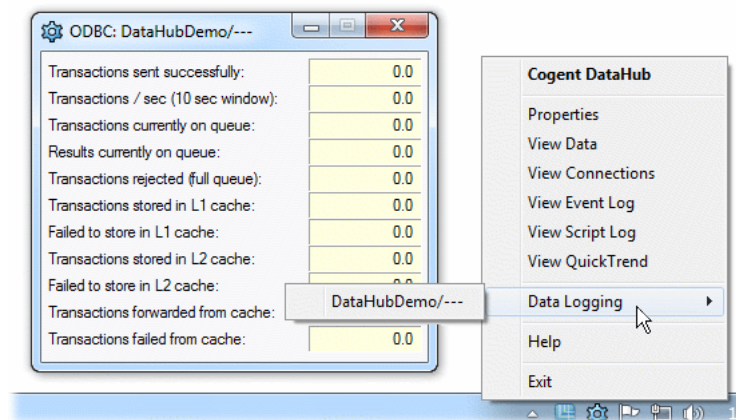
Allow duplicates while forwarding stored data

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:

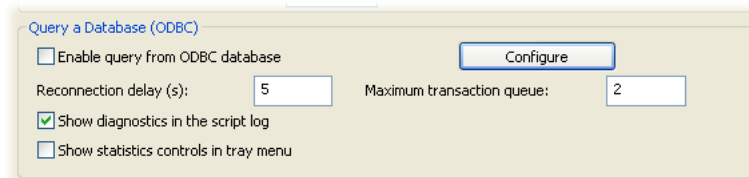
The path and directory name for the cache.

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

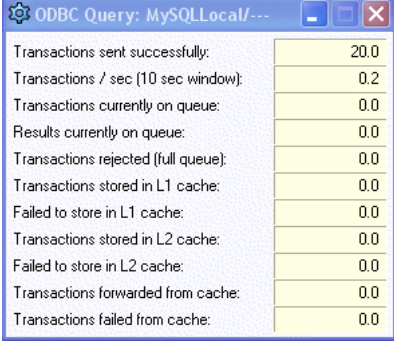
The DataHub maintains an in-memory queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. You can modify the depth of this queue to reduce the chance of involving the disk during busy periods. The queue depth for queries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

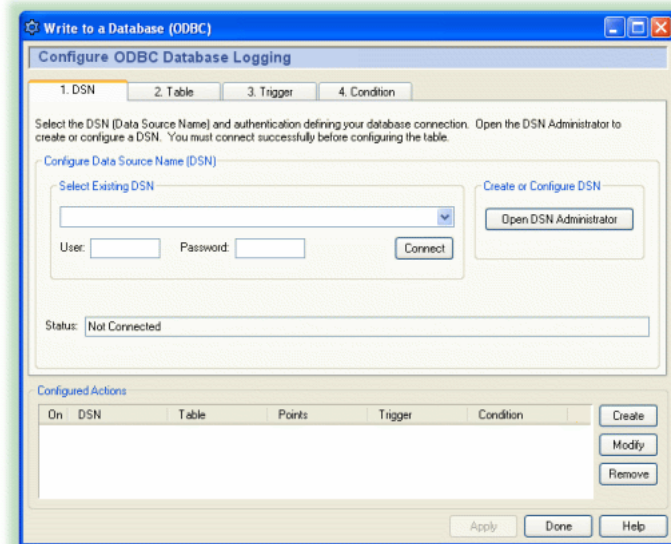
Creates a new entry in the DataHub system tray menu that lets you open a statistics control window.



Transactions sent successfully:	20.0
Transactions / sec (10 sec window):	0.2
Transactions currently on queue:	0.0
Results currently on queue:	0.0
Transactions rejected (full queue):	0.0
Transactions stored in L1 cache:	0.0
Failed to store in L1 cache:	0.0
Transactions stored in L2 cache:	0.0
Failed to store in L2 cache:	0.0
Transactions forwarded from cache:	0.0
Transactions failed from cache:	0.0

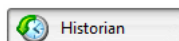
Configure button

The Configure button opens the ODBC Data Logging Window.

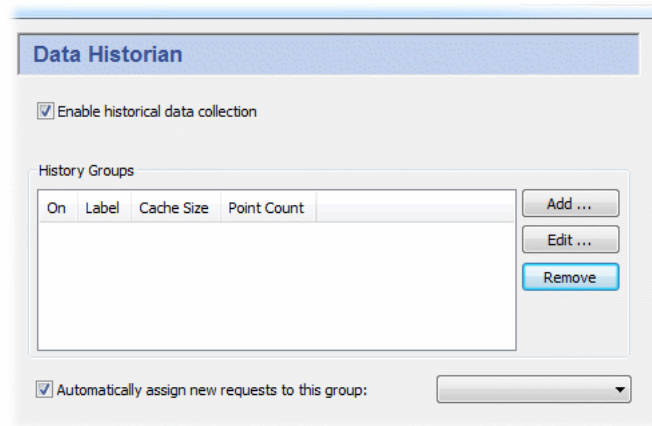


For detailed instructions on using this interface, please refer to [Chapter 10, Query a Database](#).

20.11. Historian



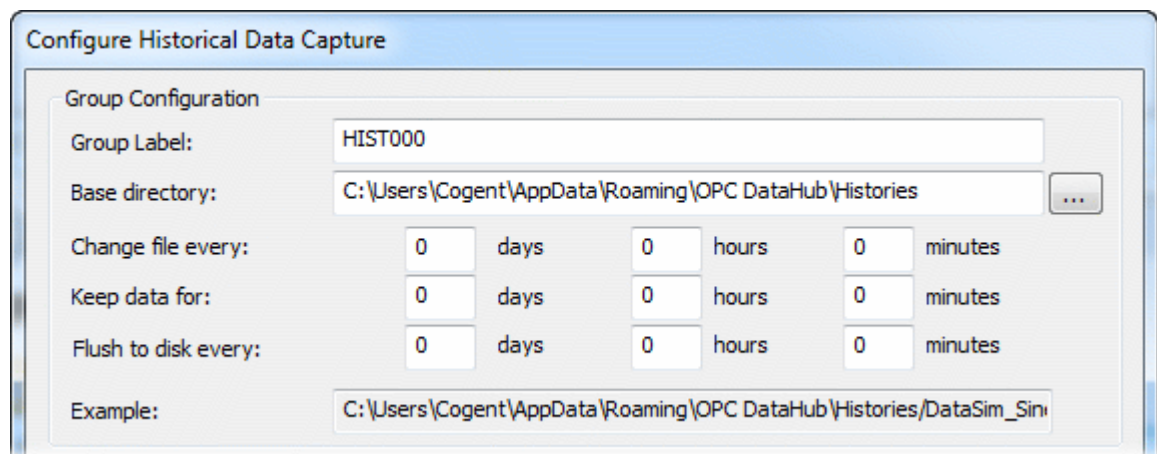
The [Historian option](#) allows you to collect and store histories for groups of data points. It gets configured automatically by the [Quick Trend](#) option, and can be configured manually as well.



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.

Deadband

☐ Absolute change: 0

☐ Percent change: 0 %

☐ Maximum time between values: 0 seconds

☐ Maximum number of skipped values: 0

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.

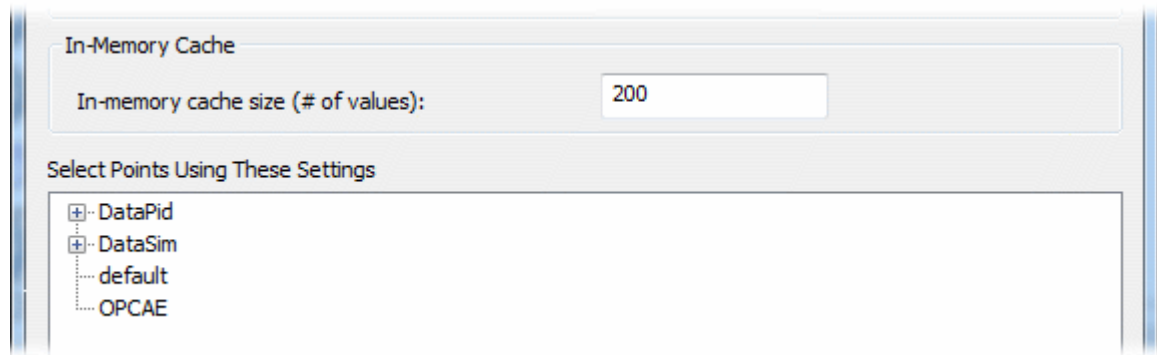
Maximum time between values:

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

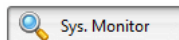
Any number of points can be selected for the group, using the selection tree.

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

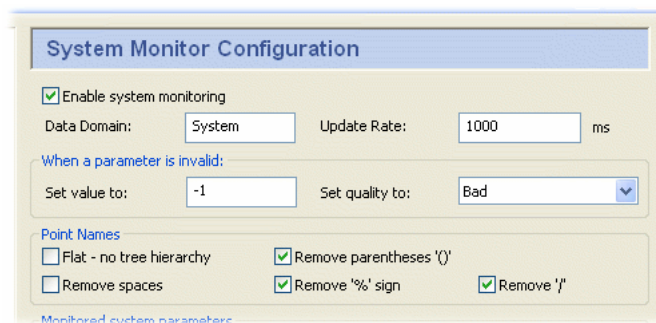
It is possible to query the Historian to extract raw data as well as statistics like minimums, maximums, averages, variances, standard deviations and so on over a given time period. This is done with a script that accesses the `Historian` class, explained in detail with an example in the DataHub Scripting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

20.12. 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.



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.



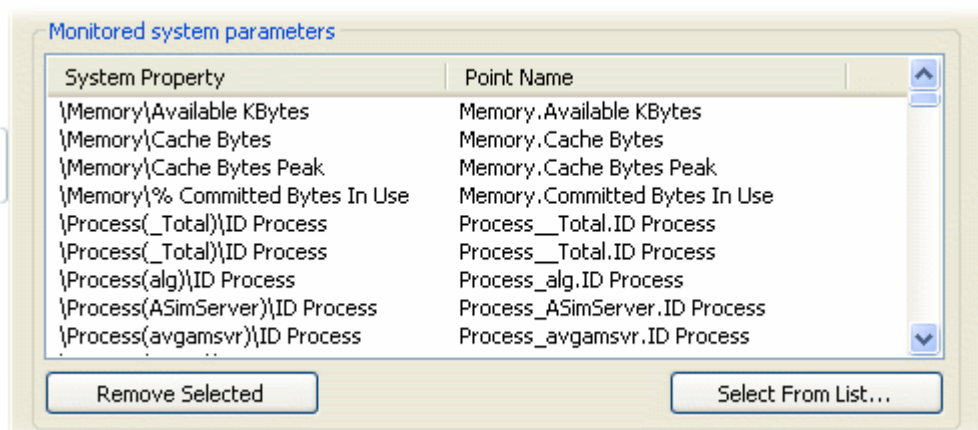
A high update rate (a low number here) for many data points could use a great deal of CPU.

When a parameter is invalid

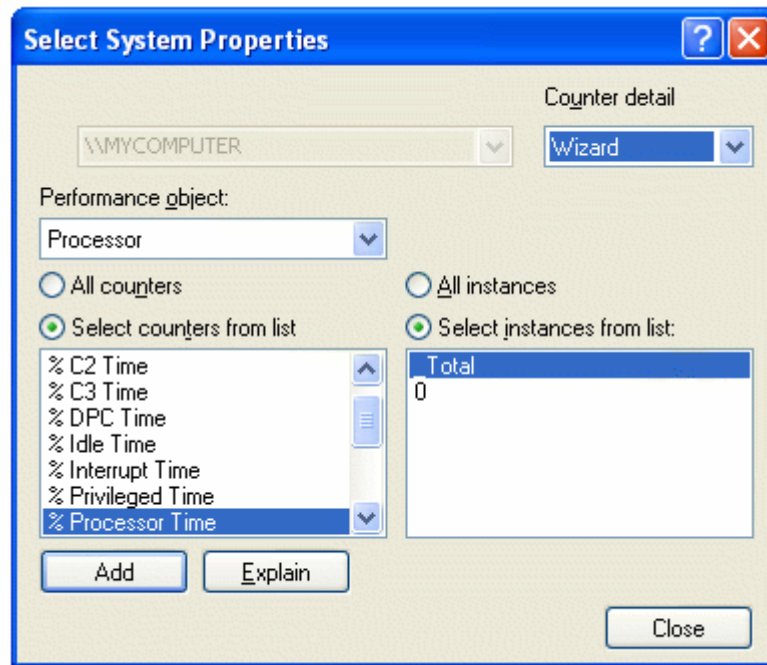
A parameter will be invalid if the object being monitored is not available. For example, if a process is not running then the parameters for that process will all be invalid. This is a useful way to monitor a system process or other object. For example, you could use a script or other client to watch 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.

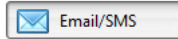
A number in this list normally indicates a selection from multiple objects of a given type, and `_Total` means the total across all of the objects. For example, if you are looking at `Processor` in a multi-processor machine, you will see a number (0, 1, etc.) for each processor and a `_Total` for the 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 [Section 18.1, Data Points](#) for more information on creating and deleting points.

20.13. 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.

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

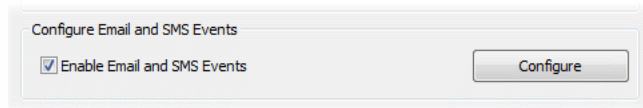
Allows the DataHub to choose the most secure type of connection available on the mail server.

Accept invalid or untrusted certificates

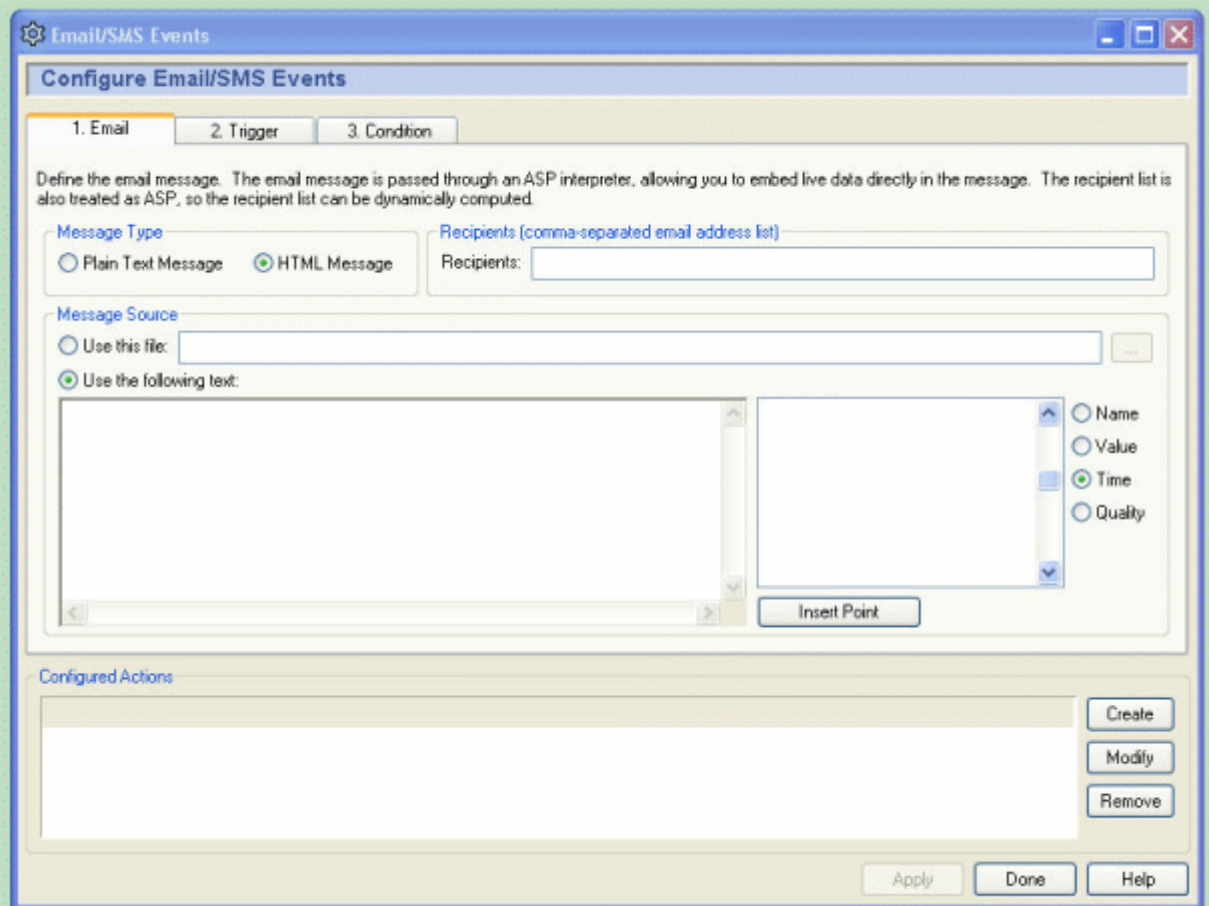
Ignores security certificate warnings.

Click the OK button to submit your entries.

Configure Email and SMS Events



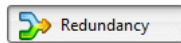
To Configure Emails press the Configure button to open the Email/SMS Events window:



A complete explanation for this window can be found in [Chapter 12, Email and SMS](#), starting with the [Sending a Test Message](#).

Click the OK button to submit your entries.

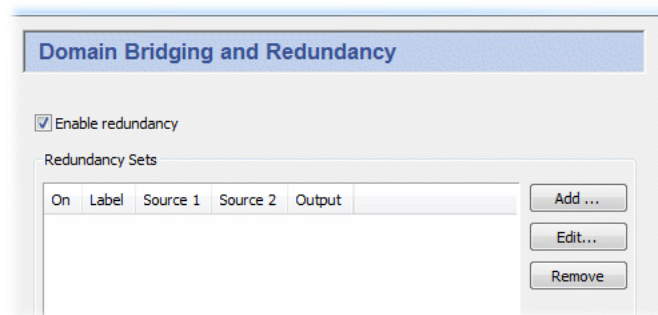
20.14. Redundancy



The [Redundancy option](#) lets you configure redundant connections to the Cogent DataHub.

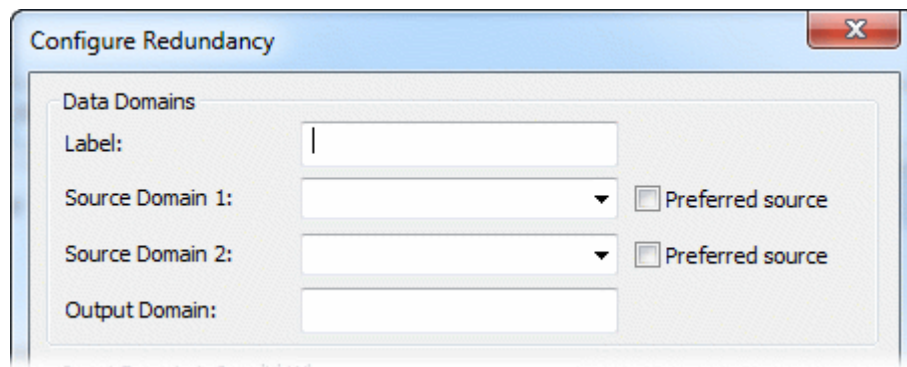
The purpose of redundancy is to collect data from two data sources and present to the client program a single output data set. The Cogent DataHub will determine which source will be presented to the client program, and switch between the two sources without affecting the client. The client will only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



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:



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:

Bad	EGU Exceeded	Last Usable	Sensor Calibration
Comm Failure	Good	Local Override	Sensor Failure
Config Error	Initializing	Not Connected	Sub Normal
Device Failure	Last Known	Out of Service	Uncertain



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

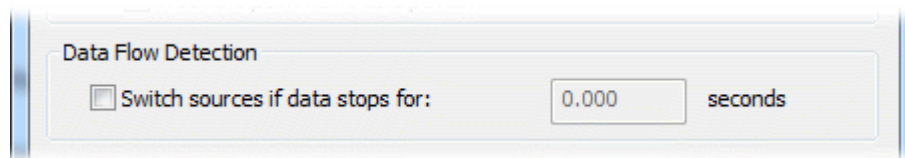
Lets you monitor all points in the domain and switch when any one of them meets the criteria.

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.

Data Flow Detection

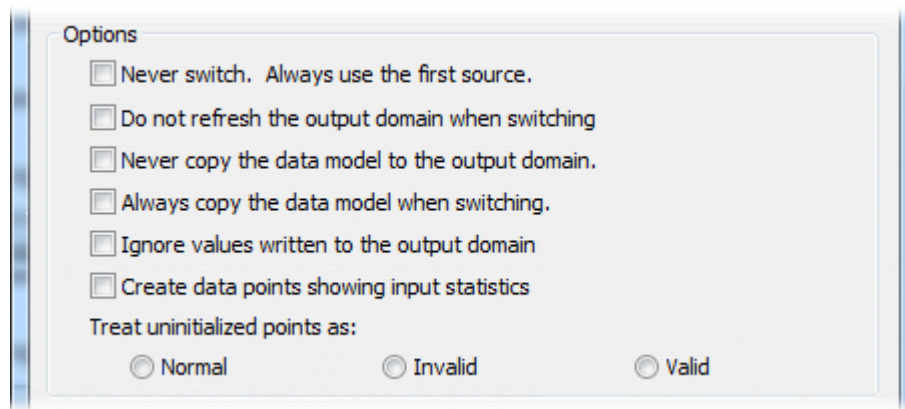
For data that changes regularly, Data Flow Detection lets you switch data sources whenever a gap in the data flow is detected. This option will watch for any change in any data point from an input domain. If any point in the input domain changes within the specified time, the entire input domain is assumed to be available. A data change must also pass the validity checks above in order 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.

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

Status and Control Data Points (blank for disabled)

Status and Control Data Points (blank for disabled)

Point for current source number:

Point for current state of domain 1:

Point for current state of domain 2:

Point for preferred source number:

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.

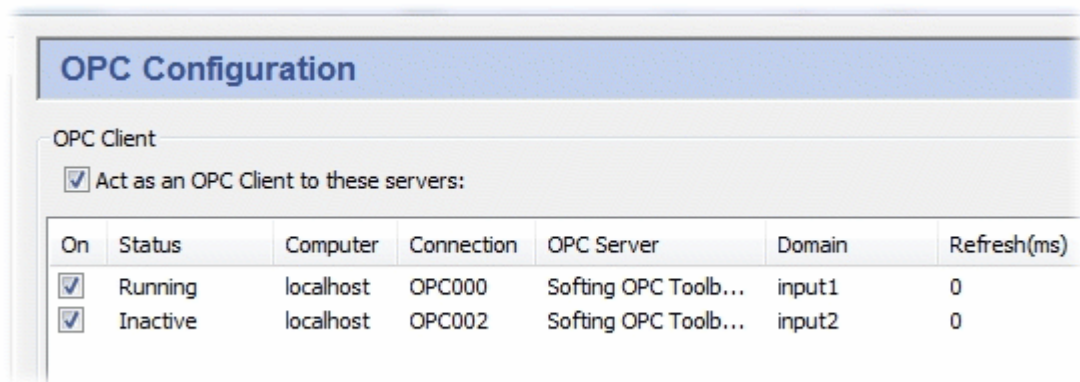
Click the OK button to submit your entries.

20.14.1. Warm Standby

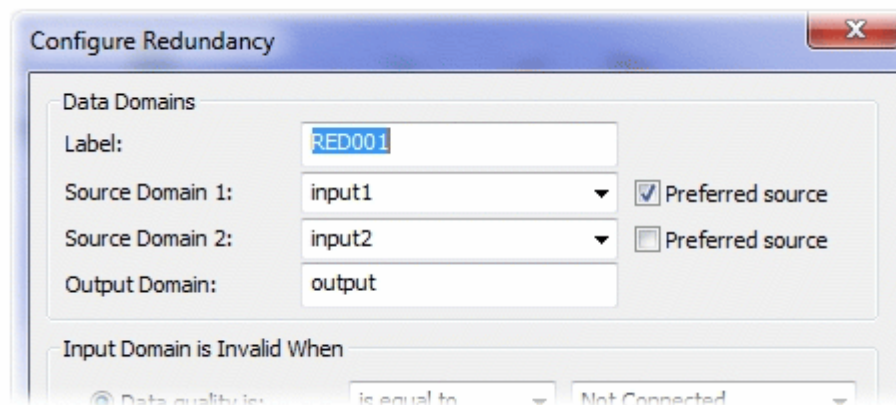
It is possible to use the DataHub's Redundancy feature to support warm standby, by using a DataHub script. Since we perform redundancy at the data level instead of the connection level, the DataHub does not naturally know which connections to start and stop when it switches input domains. Consequently 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.



- This redundancy pair *must* include configuration for the four status points. You can name them whatever you like:

Status and Control Data Points (blank for disabled)	
Point for current source number:	CurrentSource
Point for current state of domain 1:	State1
Point for current state of domain 2:	State2
Point for preferred source number:	Preference

- 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.

```

1
8  method WarmStandby.constructor ()
9  {
10     local redconf = new DomainBridge();
11     redconf.SetApplication(self);
12     redconf.Attach("RED001", "input1", "input2", "output", 0, 0, nil, 0,
13                   "CurrentSource", "State1", "State2", "Preference");
14     redconf.OPCWarmStandby("OPC000", "OPC002");
15 }
16

```

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.

WarmStandby.g example script

```

require ("Application");
require ("DomainBridgeSupport");

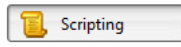
class WarmStandby Application
{
}

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);

```

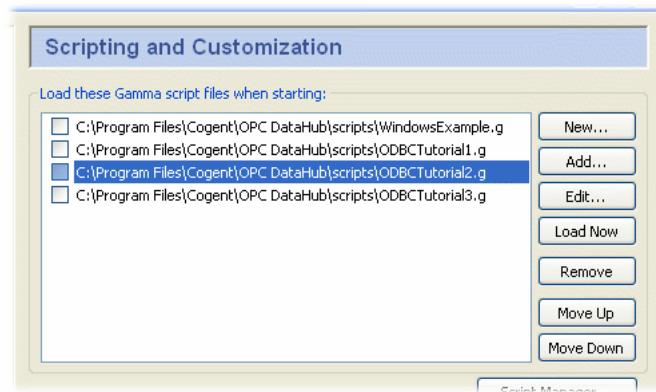
20.15. Scripting



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.

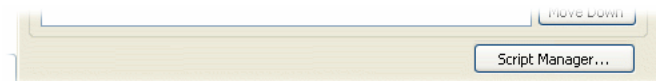


The DataHub Scripting manual has a more complete explanation for these options, but briefly they are as follows:

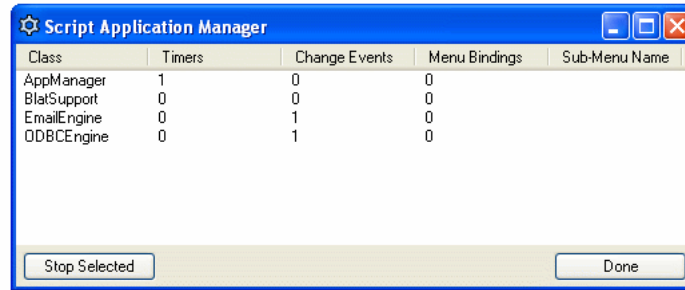
- New creates a new script from parameters you specify.
- Add lets you add a script to the list.
- Edit opens the script in the Script Log for editing.
- Load Now loads the selected script into the Gamma interpreter for immediate processing.
- Remove deletes the selected script from the list.
- Move 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.

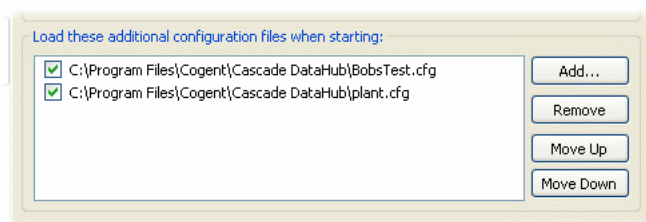
Load these additional configuration files when starting:

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 [Section 1.6, 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.



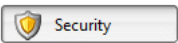
Creating or editing configuration files should only be attempted by experienced users.



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.

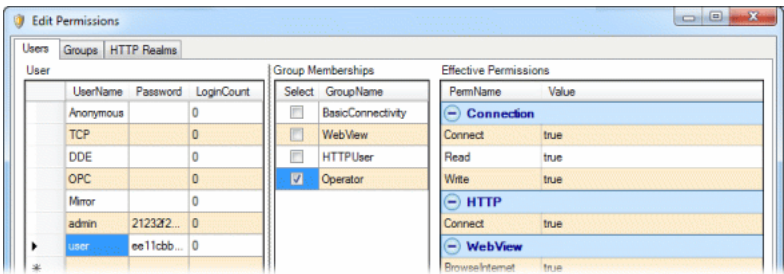
20.16. Security



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 [Chapter 17, Security](#).



Click the Configure Permissions button to open the Edit Permissions window.

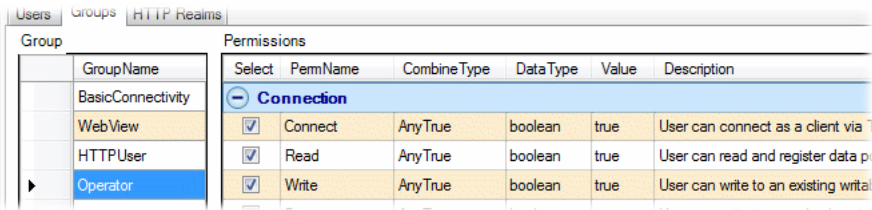


Here you can create and modify groups, and then assign users to those groups.

Groups

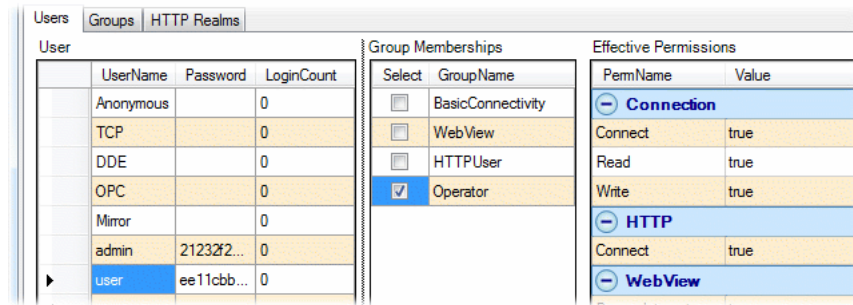
Groups provide a convenient way to configure a number of users who have identical permissions. Each group can be assigned a unique set of permissions from the **Permissions** table. There are three default groups: **Permissive**, **WebView**, and **HTTPUser**. To add a group, type a group name in the 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**.

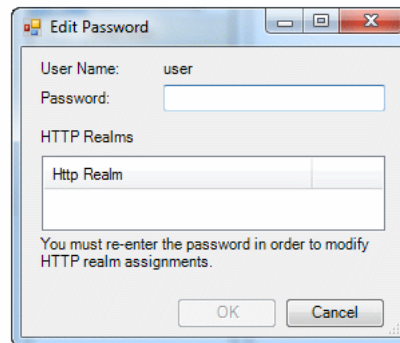


Users

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 [Section 17.3, User Authentication](#).



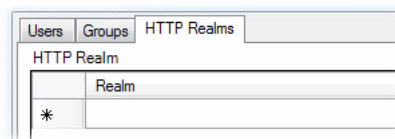
To add a user, type a user name in the bottom row of the **User** table. When you press **Enter**, a password dialog will appear:



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 [Section 17.6, Passwords](#).

HTTP Realms

Here you can maintain a list of HTTP authentication realms. This list is accessed by the DataHub Web Server, as described here: [Section 20.9, 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 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 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 **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 **cf** 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.



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 `cdh`.

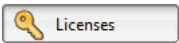
Change auto domain creation

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.

20.17. Licenses



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, the Cogent DataHub switches to license mode, and each connection then requires a license.

License Configuration			
TCP Link Licenses			
In Use:	2	Total:	5
Available:	3	Peak Use:	2
WebView Client Licenses			
In Use:	0	Total:	10
Available:	10	Peak Use:	0
Node Licenses			
Unlimited TCP Link License:	no	Unlimited Tunnel/Mirror Master:	yes
Unlimited DDE License:	yes	Unlimited OPC License:	yes

TCP Link Licenses

Each simultaneous TCP connection (except for the Tunnel/Mirror Master) to the DataHub requires one TCP Link License.

In Use

How many TCP Link Licenses are currently being used.

Available

How many are not being used and are still available.

Total

The total number of TCP Link Licenses installed.

Peak Use

The highest number of TCP Link Licenses that have been used since the beginning of the current session.

DataHub WebView Client licenses

Each simultaneous DataHub WebView client connection requires one DataHub WebView Client License.

In Use

How many DataHub WebView Client licenses are currently being used.

Available

How many are not being used and are still available.

Total

The total number of DataHub WebView Client licenses installed.

Peak Use

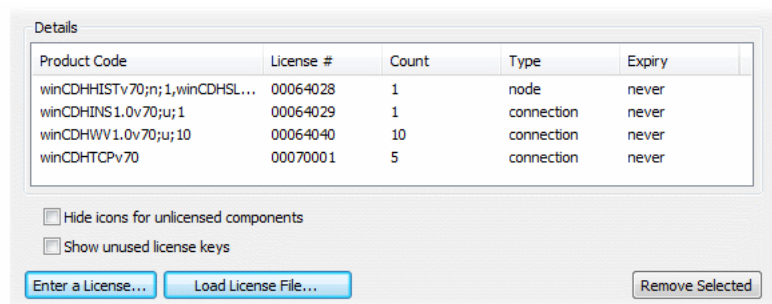
The highest number of DataHub WebView Client licenses that have been used since the beginning of the current session.

Node Licenses

This box displays which unlimited licenses are currently available for the Cogent DataHub: TCP, DDE, Tunnel/Mirror, and OPC.

Details

The Details list shows the licenses currently installed.



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.

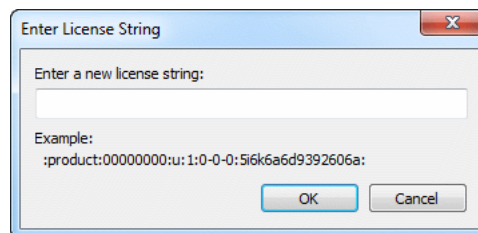
Show unused license keys

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 `l` and `1` 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 [Section 1.6, Configuration Files](#) for more information on license file locations.)

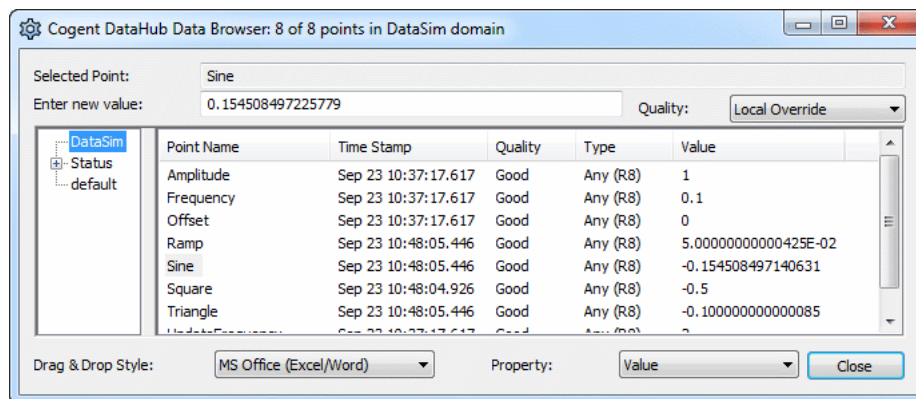
To remove a license from your system, select one or more licenses in the Details window, then click the Remove Selected button.

Chapter 21. Other Windows and Programs

21.1. 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.

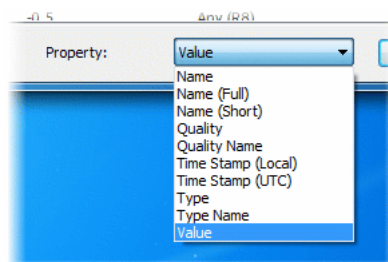


All data domains in the DataHub are shown in the tree on the left, and all the points of the selected data domain are listed on the right. Clicking on a point name selects it, and puts its name into the **Selected Point** field at the top of the window. A snapshot of the value of the point at the time 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

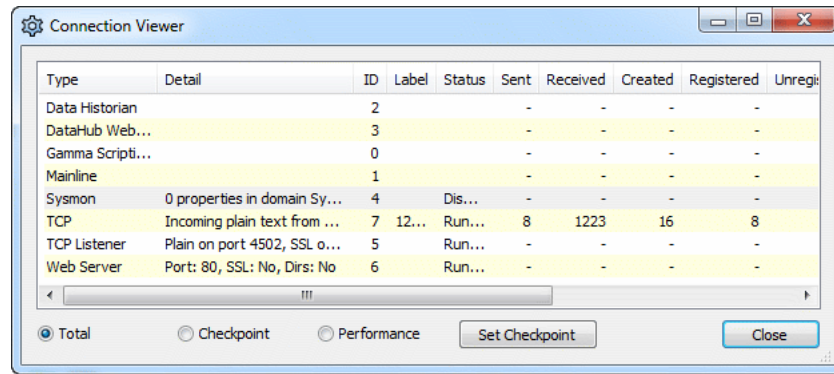
A text string that corresponds to the integer value of the data type.

Value

The value of the point.

21.2. 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.

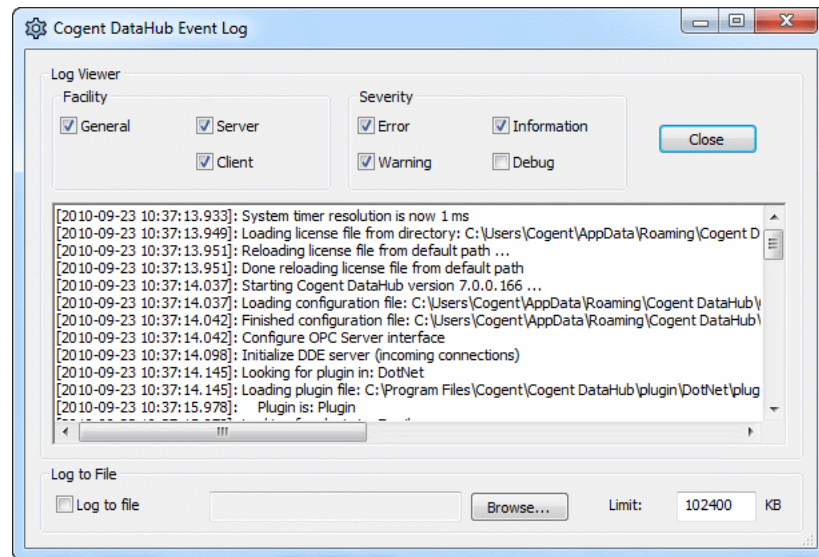
You have the following options for viewing the data:

- **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.

21.3. 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 and 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.

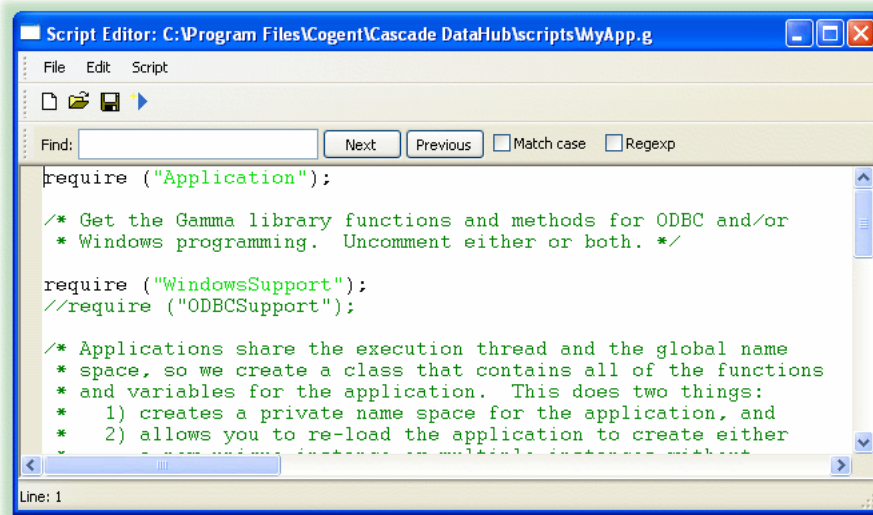
Here are some other useful pointers about log file management:

- 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.

- Checking the Log to File button automatically causes the Event Log to log in the (verbose) Debug mode.

21.4. Script Editor

The Script Editor¹ lets you write and edit scripts.

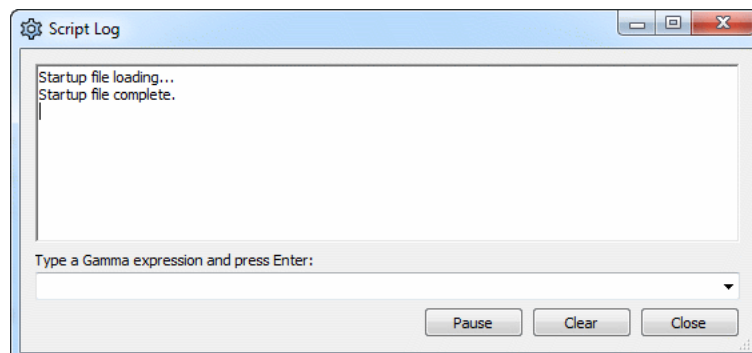


The Script Editor offers features such as context-sensitive highlighting, prompted fill-ins for functions and variable names, automatic indenting, text string searches, and so on. Please refer to the DataHub Scripting manual for detailed information.

21.5. 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:

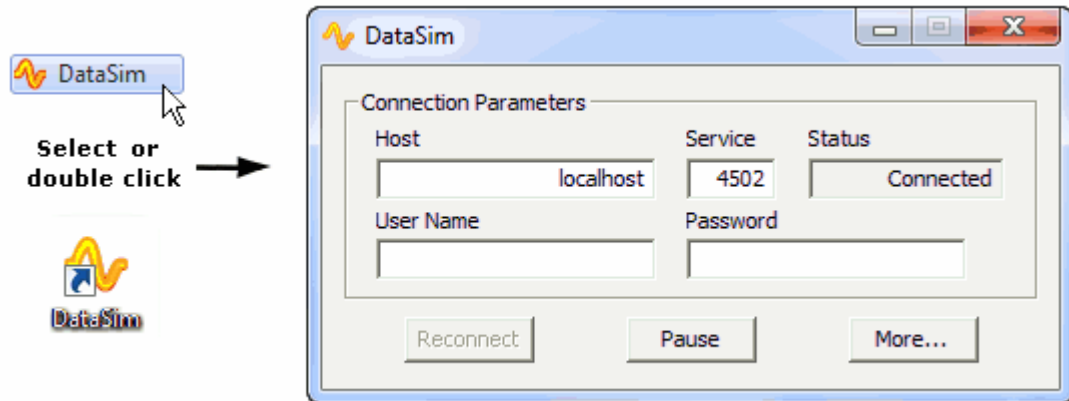
- Click the Script Log button in the Properties window.
- Right click the DataHub icon in the system tray and select View Script Log from the pop-up menu.



Please refer to the DataHub Scripting manual for detailed information about the Script Log.

21.6. 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.

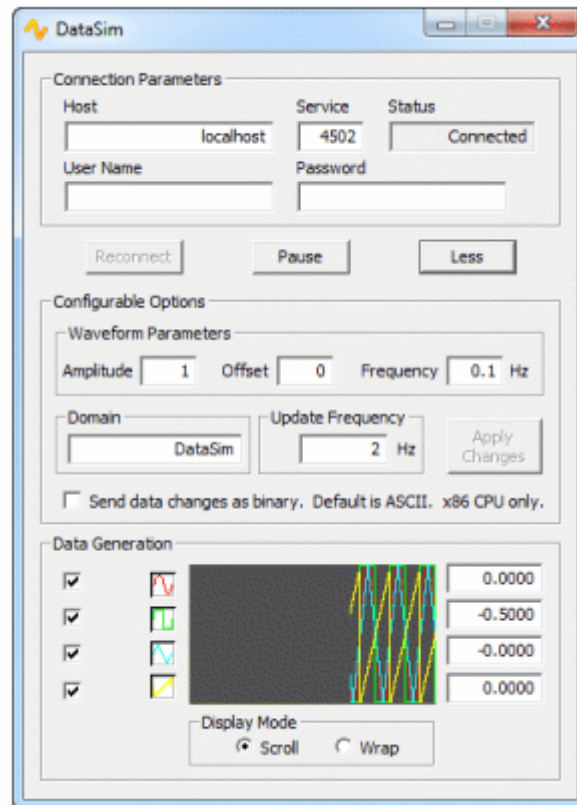
Service

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 button 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.

21.7. DataPid - a PID loop data simulation program

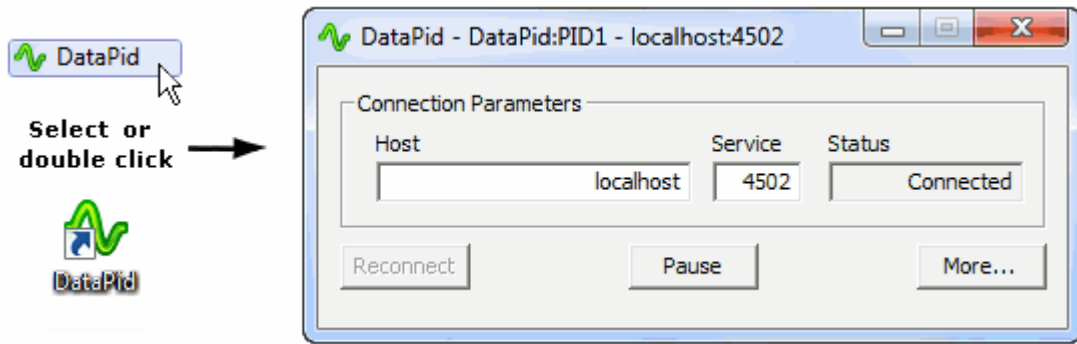
DataPid is a data simulation program for the Cogent DataHub. It simulates data for a set point, control output, and process variable, and sends these values to the DataHub by a TCP connection.



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 set up 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.

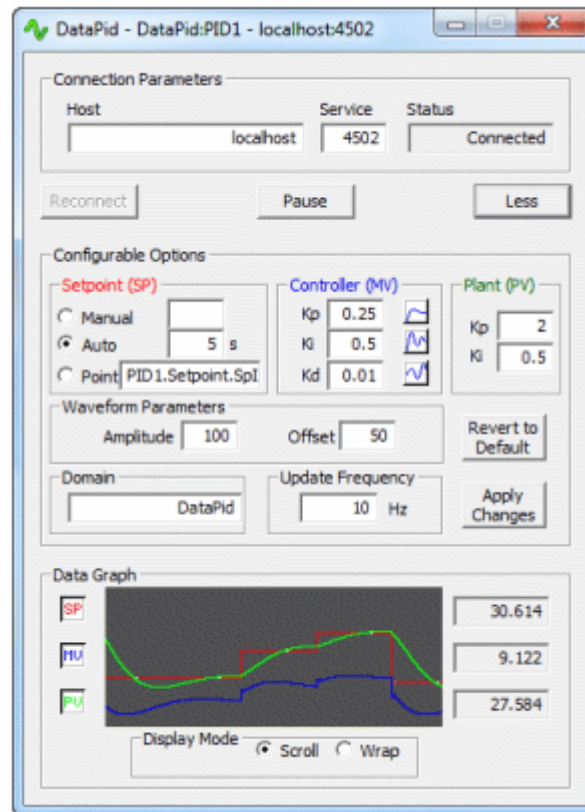
Service

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 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

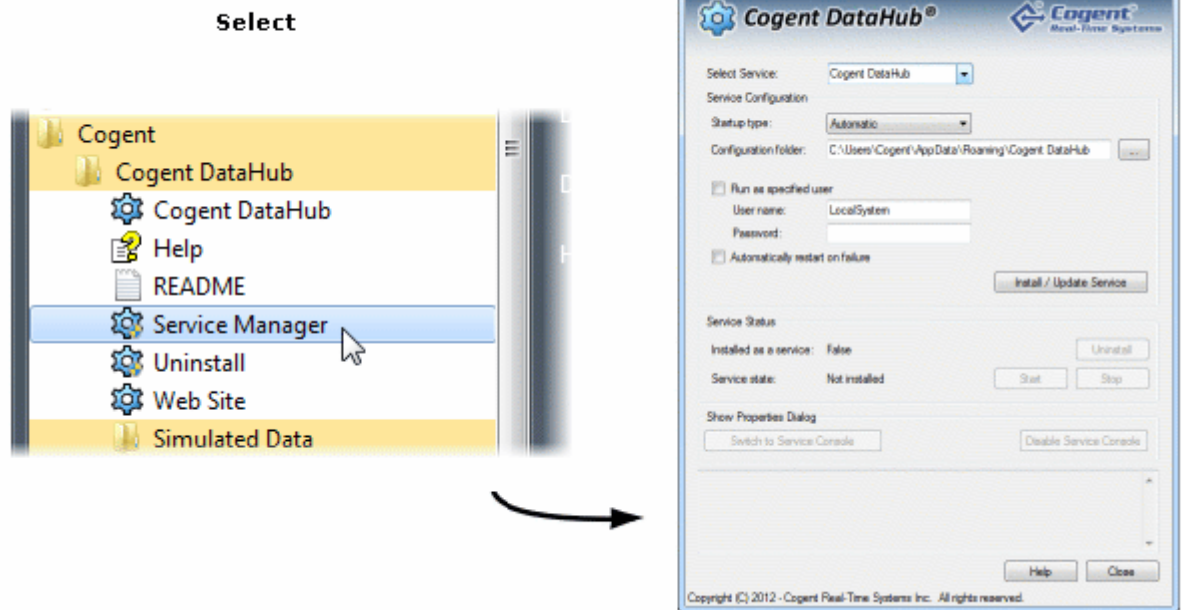
Controls how the data is displayed in the trend display. **Scroll** moves the wave from right to left as the data is generated. Old data scrolls off the left side of the display as new data scrolls in from the right. **Wrap** adds to the wave from left to right, and writes over old data.

21.8. 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.

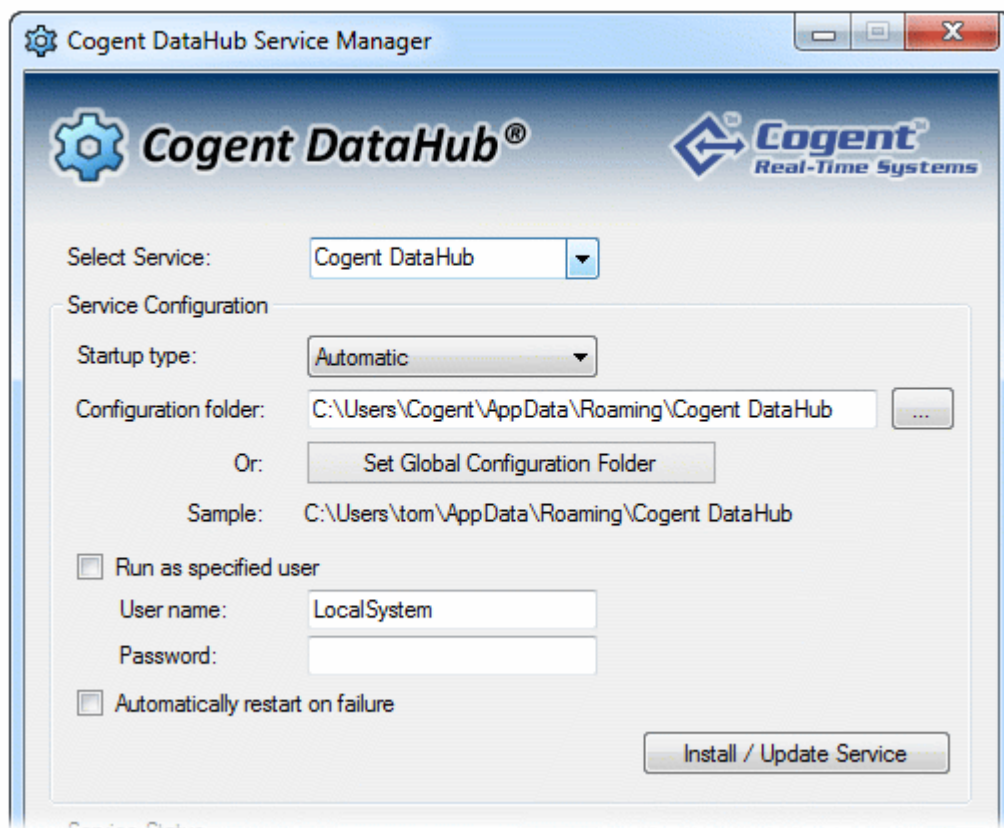
Starting the Service Manager

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 [Section 1.6, Configuration Files](#) for more information about configuration files.

The **Set Global Configuration Folder** button lets you specify the configuration file through the registry. This will preserve the configuration directory even after uninstalling as a service.

Run as specified user

It is recommended that you run the DataHub as the local SYSTEM (LocalSystem) user, and leave this box unchecked.



The DataHub Properties window is only available when running the service as the local SYSTEM user. If you run as a specified user then you will not be able to access the Properties window to make changes to the DataHub while it is running as a service.



However, there are certain situations in which you may need to run it as a specified user. To specify a user other than the local SYSTEM user, enter the **User name** and **Password** as applicable, then please see [Appendix C, Running as a Windows Service \(Specified User\)](#) for important additional information

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**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 2003

A Show Properties Dialog button allows you to open the Cogent DataHub Properties window.

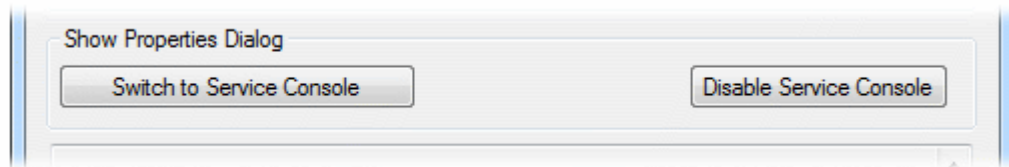


The DataHub Properties window is only visible on the primary console of the computer running the Cogent DataHub. If you are currently logged in via a remote desktop session, you will see a pop-up dialog indicating that you must be connected to the computer's primary console. You can do this by using the `/admin` or `/console` options on the Microsoft Remote Desktop client.

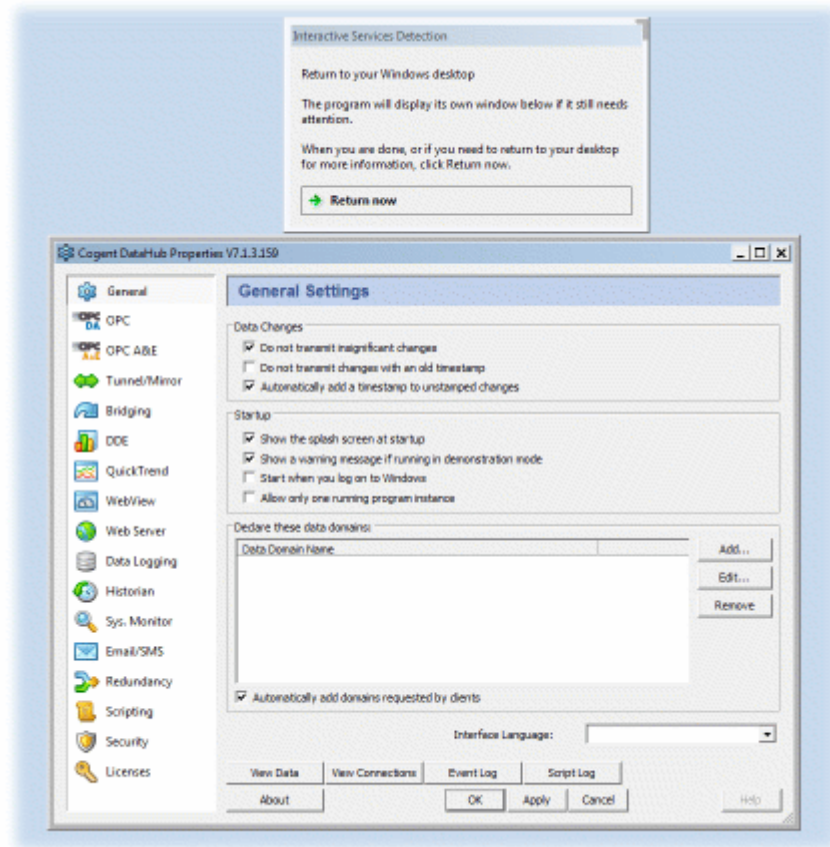
If you are already connected to the primary console, the DataHub Properties window and system tray icon will be displayed when you press Show Properties Dialog button.

Windows Vista, 7 and 2008

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.



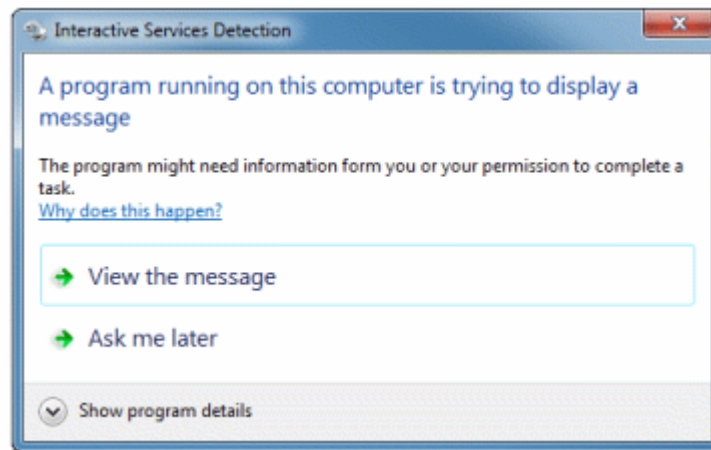
While the Service Console is open, your other applications will continue to run normally, but you will not be able to see or interact with them. A dialog box will be visible while you are viewing the Service Console that will allow you to switch back to your regular desktop at any time. If you have other 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 the DataHub properties, before returning to the normal user console, you should click the Apply and OK buttons to close down the 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.

Notes

1. This editor is based on the Scintilla and SciTE editor.

Chapter 22. Using DataHub Commands

The DataHub has an internal command set, documented in the [Reference I, 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 [Section 1.6, 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.

22.1. 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.

22.2. 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_WOULDBLOCK	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.

22.3. Sending Commands by TCP

It is possible to send commands directly to the DataHub over TCP, by using a [tunnelling/mirroring](#) connection. The communication between a client and the DataHub over TCP follows these guidelines:

- 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 bit with writing lisp expressions. They automatically add double-quotes around `%s` formatted strings, and escape characters within the string.

Chapter 23. Troubleshooting

This chapter has answers to the most common troubleshooting questions that may arise. Any other known, unresolved issues are discussed in the `README.txt` which is included in the distribution archive. Please read that file and this page before contacting Cogent for help.



For responses to other common questions and issues, please see also the Cogent Community Forum Frequently Asked Questions.

1. I can't get a connection.

Check the following:

- Is your connection to the Internet and/or network up and functioning normally?
- Does your firewall allow a local program (like the Cogent DataHub) to act as a server?
- Does the Cogent DataHub [run](#) OK?
- Is the DataHub [installed and configured](#) correctly?
- For drag and drop operations, is the Drag & Drop Style at the bottom of the Data Browser set to MS-Office (Excel/Word)?
- Are you using the DataHub [web server](#) and is Skype installed on the same machine? There is a default option in Skype (follow the Tools - Options - Connection menus) to use port 80 as an alternative for incoming connections. You will need to either disable this, or change the port number of the DataHub's web server to something other than 80.

2. Windows Error Messages

For a list of Windows error messages, including those for TCP and DDE, please refer to [Appendix H, Error Messages](#).

3. Cogent DataHub Error Messages

Here is a brief explanation of some of the more common error messages you might encounter when using the Cogent DataHub:

License failure: License in use by master

There are two possible reasons for this message:

1. One Cascade DataHub is tunnelling/mirroring to another DataHub on another computer and that DataHub is tunnelling/mirroring back to the first.
2. The Cascade DataHub is tunnelling/mirroring to itself.

Either of these are probably due to misconfiguration. Check your configuration files to ensure that only one Cascade DataHub is configured to be a master in any master/slave pair, and that no DataHub is tunnelling/mirroring to itself.

4. 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.

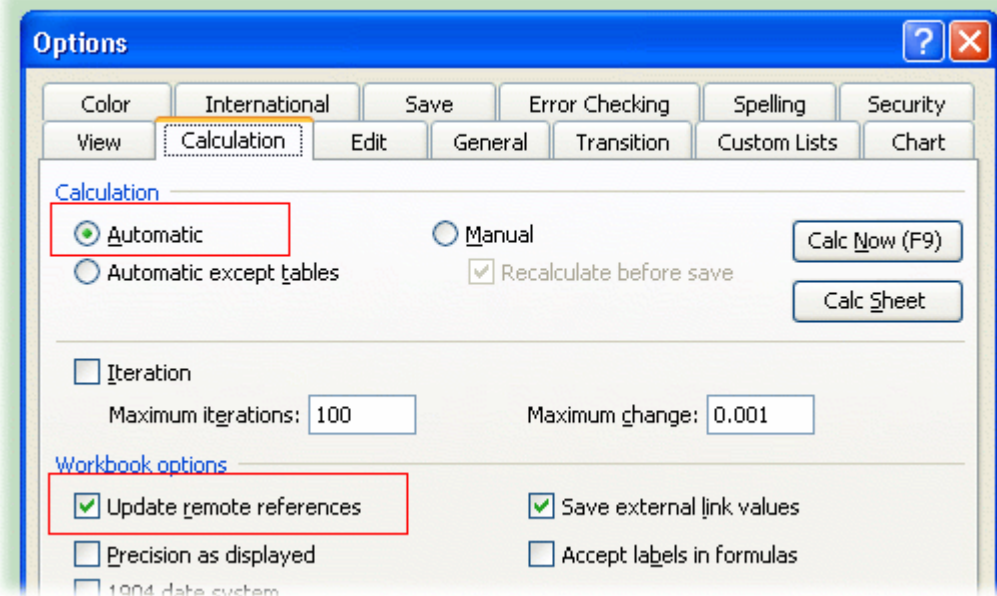
Remote data not accessible. Start the DataHub?

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.

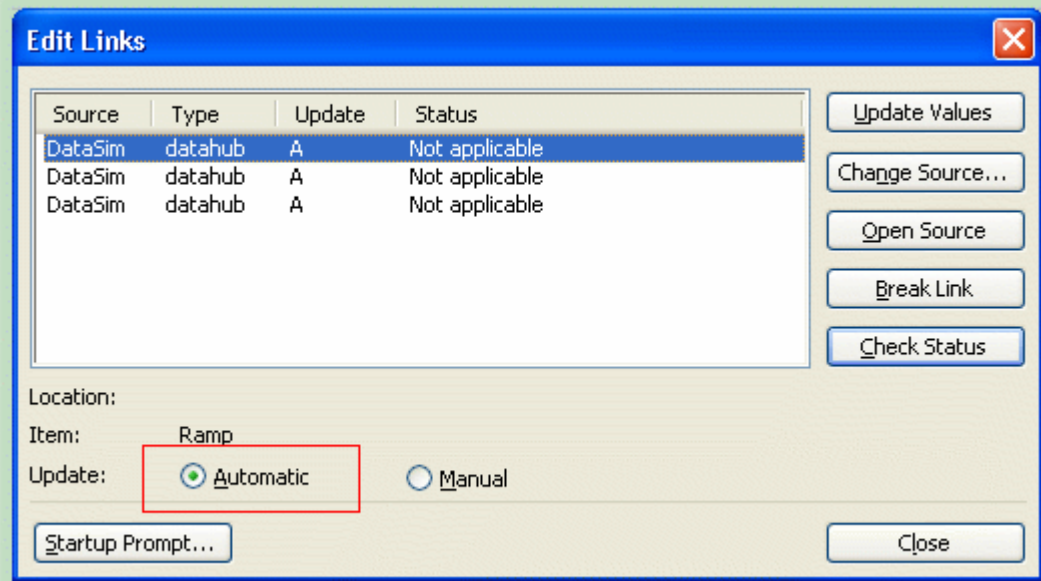
5. Excel not updating

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. From the **Tools** menu, choose **Options** to open the Options window.



2. Ensure that the Automatic option in Calculation is selected.
3. Ensure that the Update remote references option in Workbook options is selected. Then close the Options window.
4. From the **Edit** menu, choose **Links** to open the Edit Links window.



5. Ensure that the Automatic option for Update is selected. Then close the Edit Links window.

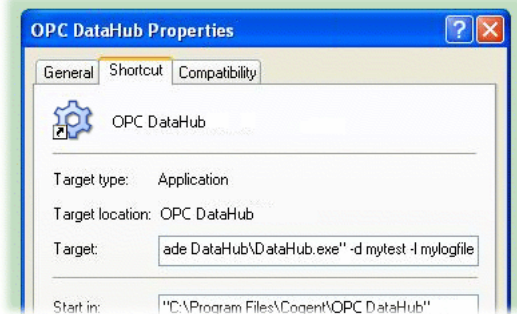


For responses to other common questions and issues, please see also the Cogent Community Forum Frequently Asked Questions.

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 Cascade 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 -U. If the directory cannot be found or created, the files will be stored in the installation directory.

-l *file*

Log messages to this file.

-m *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;

-M *address*

Acting as a TCP slave, attach to a TCP master on this host. The address is a machine name, such as `developers.cogentrrts.com` or a machine address, such as `192.168.3.15`.

-n *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.

-p *port*

Act as a TCP master and listen on this port/service.

-q *queue*

Specify an alternate queue name for this DataHub. Normally **datahub** chooses its own queue name to be unique on the network.

-s

Synchronized: The DataHub will ignore changes to a point if the point's current timestamp is more recent.

-t

Automatically generate a timestamp on unstamped points.

-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 `-H`.

-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.

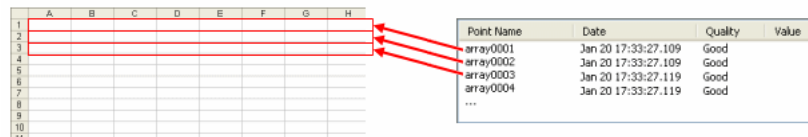
Appendix B. Excel Macro Library

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.

B.1. 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.

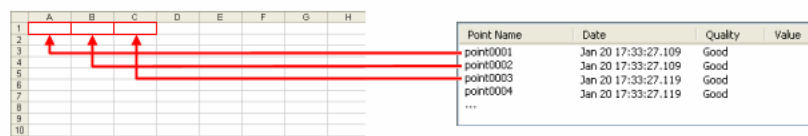


Each data point represents a row of data. This macro assumes the names are "array0001", "array0002", etc.

```
-----
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
-----
```

- **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
-----
```



```

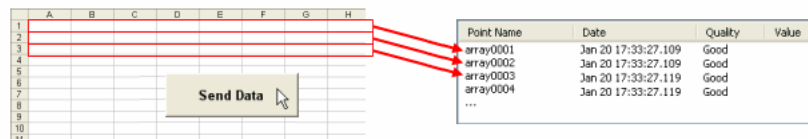
Next i
End Sub
-----

```

B.2. 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.

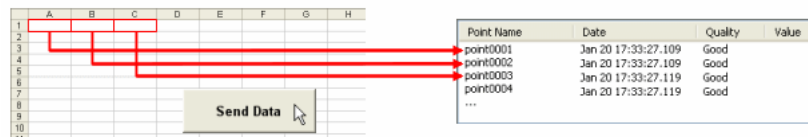
```

-----
Sub transmit_arrays()
    Dim chan As Integer
    Dim pname As String

    chan = DDEInitiate("datahub", "default")
    For i = 1 To 100
        pname = Format(i, "0000")
        pname = "array" & pname
        DDEPoke chan, pname, Worksheets("Sheet1").Range(Cells(i, 1), Cells(i, 40))
    Next i
    DDETerminate (chan)
End Sub
-----

```

- **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.



This macro assumes that the data points are named "point0001", "point0002", etc.

```

-----
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
-----

```

B.3. Write data from Excel - Automatically on value change (using DDEPoke)

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

```

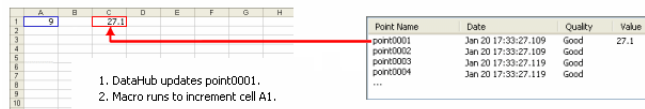
Add to Sheet1 macro code:

```
-----
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
-----
```

B.4. 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.



Point Name	Date	Quality	Value
point0001	Jan 20 17:33:27.109	Good	27.1
point0002	Jan 20 17:33:27.109	Good	
point0003	Jan 20 17:33:27.119	Good	
point0004	Jan 20 17:33:27.119	Good	
...			

Add to Sheet1 macro code:

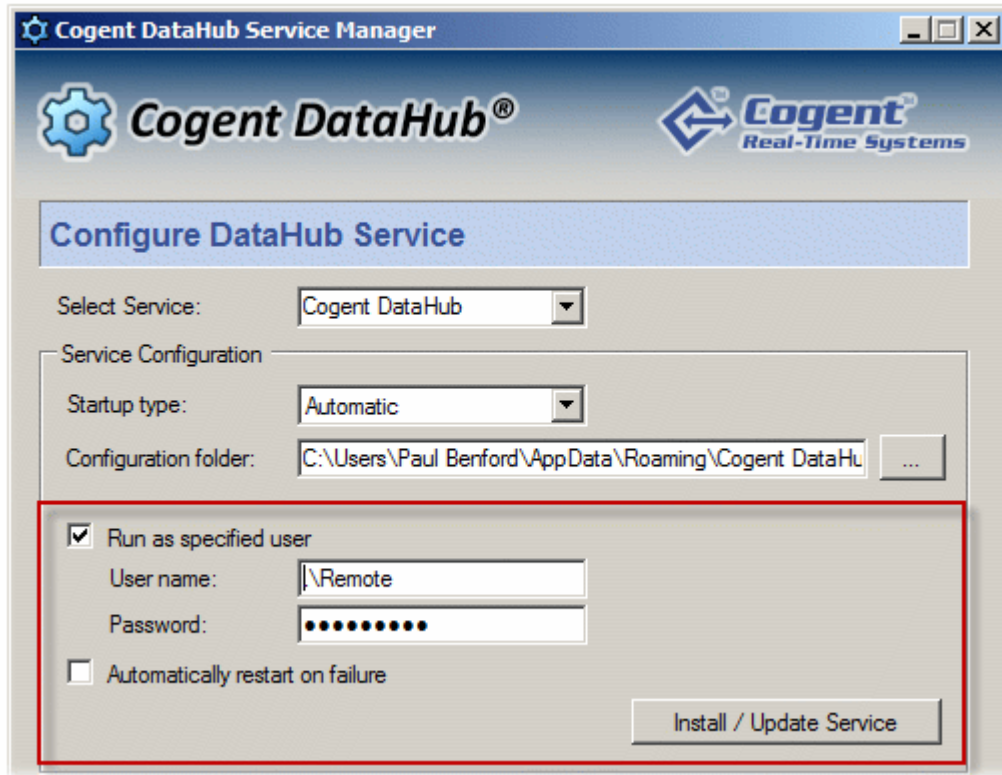
```
-----
Sub link_updated()
    Cells(1, 1) = Cells(1, 1) + 1
End Sub
-----
```

Run once to establish the link:

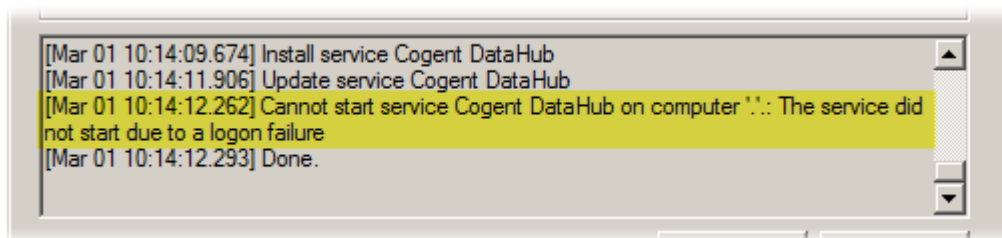
```
-----
Sub set_link()
    ThisWorkbook.SetLinkOnData "datahub|default!'point0001'", "Sheet1.link_updated"
End Sub
-----
```

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.



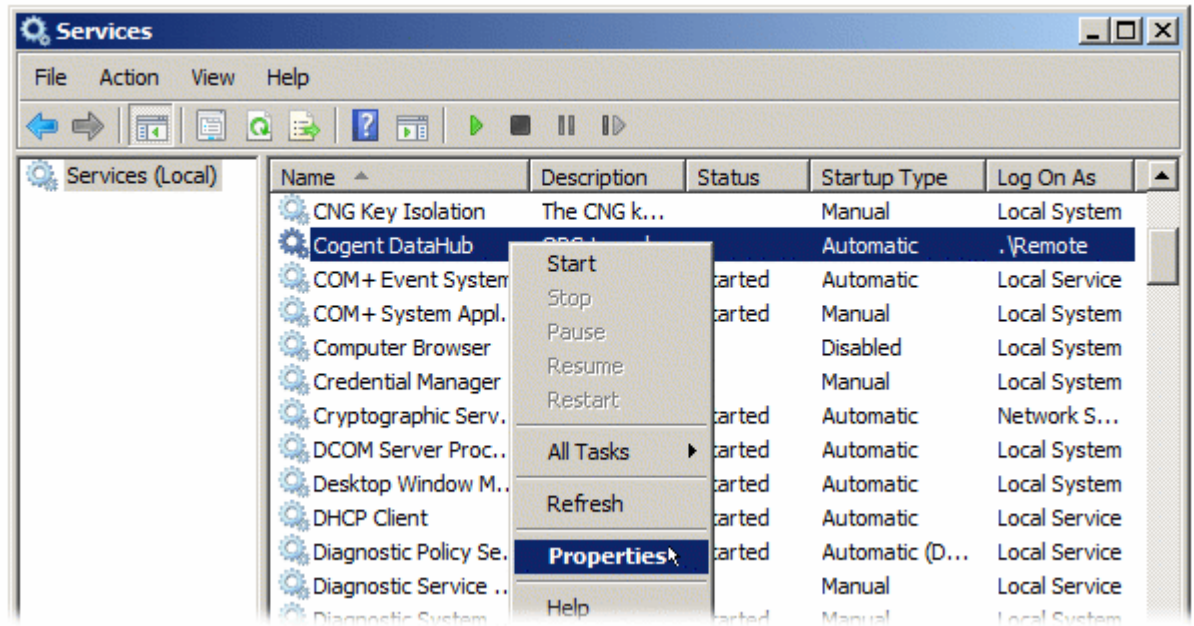
If you try to install the DataHub service as a specified user as explained in [Section 1.3, Installing the DataHub as a Service](#) then you may find that the service does not run, and you get an error message related to a logon failure, such as this:



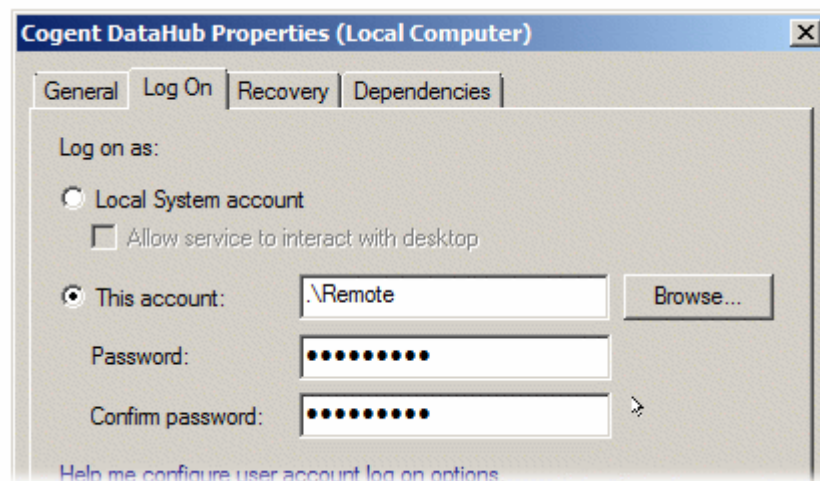
This error occurs when a user who is going to run the DataHub service does not have the permission, or right, to log on as a service in Windows.

One way to set this permission/right is as follows:

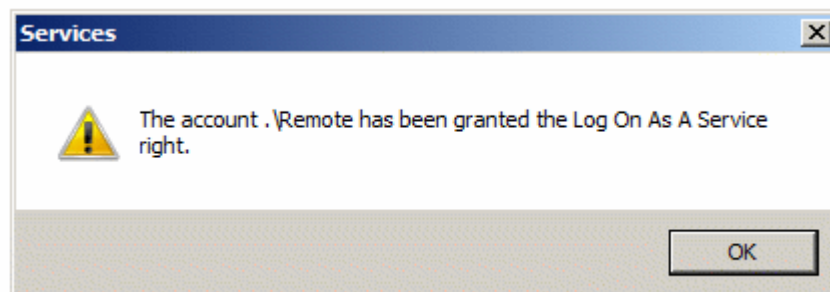
1. Open the Windows Services window from the Control Panel by selecting Administrative Tools and then Services.
2. Right click on Cogent DataHub and select Properties.



- Click on the Log On tab and then retype the password for the user account you are trying to use. Then click Apply.



- You should see a Windows message acknowledging that the right to log on as a service has been granted.



- 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 `\winnt\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
```

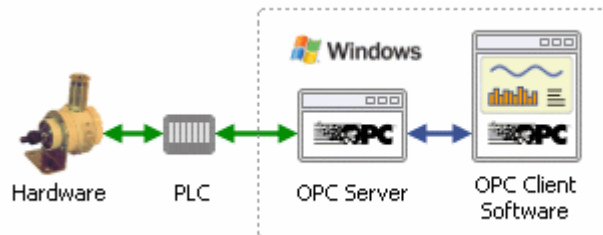
using a text editor. For example, to assign the name `datahub` to port 4502, you would add the line:

```
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. OPC Overview

OPC¹ is a software interface standard² that allows Windows programs to communicate with industrial hardware devices.



OPC is implemented in server/client pairs. The **OPC server** is a software program that converts the hardware communication protocol used by a PLC³ into the OPC protocol. The **OPC client** software is any program that needs to connect to the hardware, such as an HMI⁴. 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.

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 [OPC aggregation](#).
- Connect an OPC client to an OPC server over a network. This can be done with [OPC tunnelling](#).
- Connect an OPC server to another OPC server to share data. This is known as [OPC bridging](#).

The OPC DataHub is uniquely designed to do all of these tasks. It is a combination OPC server and OPC client that supports multiple connections. Thus it can connect to several OPC servers simultaneously, for [OPC aggregation](#) and [OPC bridging](#). Two OPC DataHubs can mirror data across a TCP network to provide [OPC tunnelling](#).

In addition to enhancing OPC server and client connections, the OPC DataHub can connect any OPC server or client to other applications as well, such as [Excel](#), a [web browser](#), or any [ODBC database](#). And it can also be used to get OPC data into [Linux](#) or [QNX](#).

Notes

1. The acronym "OPC" comes from "OLE (Object Linking and Embedding) for Process Control". Since OLE is based on the Windows COM (Component Object Model) standard, under the hood OPC is essentially COM. Over a network, OPC relies on DCOM (Distributed COM), which was not designed for real-time industrial applications and is often set aside in favor of [OPC tunnelling](#).
2. OPC actually comprises several standards, the first and most important of which is OPC Data Access (OPC DA). There are also standards for alarms & events, historical data, batch data, and XML.
3. Programmable Logic Controller: a small industrial computer that controls one or more hardware devices.

4. 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.

Appendix F. 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 *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:

Application	DDE Service Name	DDE Topic Name
Cogent DataHub	Multiple names can be assigned. The default name is <code>datahub</code> .	Any Cogent DataHub domain name, the default domain is <code>default</code> .
Microsoft Excel	EXCEL	The name of the spreadsheet, chart, macro, etc. For example: <code>mysheet.xls</code>
Microsoft Access	MSACCESS	The name of the table, SQL query, or macro to run.
Microsoft Word	WINWORD	The name of the document, including the <code>.doc</code> extension.
FoxPro	FOXPRO	
InTouch Viewer	VIEW	TAGNAME
FIX DMACS	DMDDE	DATA
National Instruments' Lookout	LOOKOUT	The name of the application, without the <code>.lkp</code> extension.

Application	DDE Service Name	DDE Topic Name
Asymetrix Toolbook	TOOLBOOK	The name of the toolbook application, with the .tbk extension.

Here are a few service and topic names for financial data feeds:

Data Feed	DDE Service Name	DDE Topic Name	Symbol Example
ADP Shark	ML	LP	IBM.N.Q
Bloomberg	BLP	M	IBM EQUITY, [LAST TRADE]
Bridge	BDDE	TKR	@USH8/LS
FXCM	FXPS	BID	EUR/USD
Future Source (CalcSource)	CALCSRC	P	USH8.LAST
Future Source (ProfNet)	PROFDDE	LIVE	APIU02, LAST
Knight Rider Profit Center	QMASTER	QUOTE	USH8.LAST
METATRADER	MT	BID	USDCHF
METATRADER v. 4	MT4	BID	USDCHF
MGFOREX	MGFOREX	RATES	USD
Moneycast	WBSEVER	SES	19/L (19 is a stock code)
Reuters	REUTER	IDN	USH8 /IBM ,LAST
Telerate WorkStation	TWINDDE	QUOTES	USH8.3 LAST
Universal Market Data Server	USDDE	QUOTE	US8H;LAST

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.

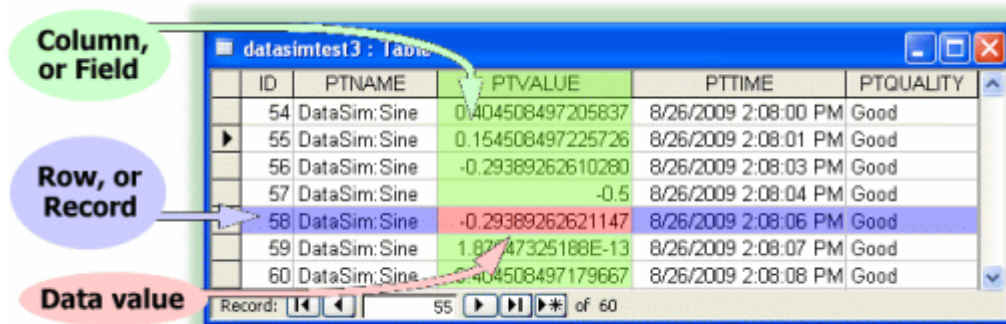
Appendix G. ODBC Database Concepts

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.

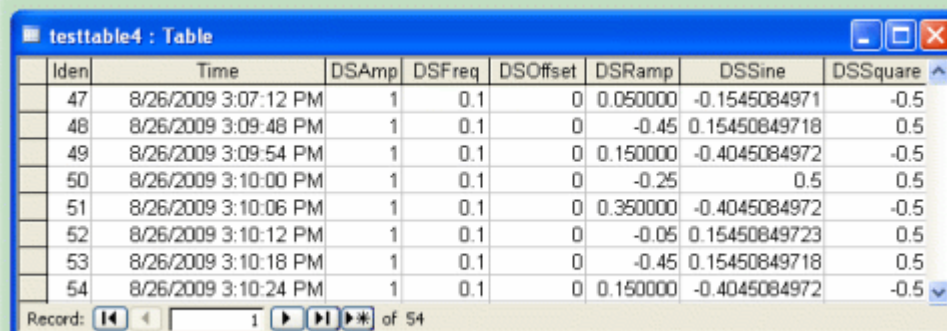
A database *table* is a logical grouping of related data, organized by the common attributes or characteristics of individual data items. Each *column* or *field* in the table contains a particular attribute, and is of one particular *data type*. Typical data types include boolean, string, numeric, date/time, 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:



ID	PTNAME	PTVALUE	PTIME	PTQUALITY
54	DataSim:Sine	0.404508497205837	8/26/2009 2:08:00 PM	Good
55	DataSim:Sine	0.154508497225726	8/26/2009 2:08:01 PM	Good
56	DataSim:Sine	-0.29389262610280	8/26/2009 2:08:03 PM	Good
57	DataSim:Sine	-0.5	8/26/2009 2:08:04 PM	Good
58	DataSim:Sine	-0.29389262621147	8/26/2009 2:08:06 PM	Good
59	DataSim:Sine	1.83747325188E-13	8/26/2009 2:08:07 PM	Good
60	DataSim:Sine	0.404508497179667	8/26/2009 2:08:08 PM	Good

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:



IDen	Time	DSAmp	DSFreq	DSOffset	DSRamp	DSSine	DSSquare
47	8/26/2009 3:07:12 PM	1	0.1	0	0.050000	-0.1545084971	-0.5
48	8/26/2009 3:09:48 PM	1	0.1	0	-0.45	0.15450849718	0.5
49	8/26/2009 3:09:54 PM	1	0.1	0	0.150000	-0.4045084972	-0.5
50	8/26/2009 3:10:00 PM	1	0.1	0	-0.25	0.5	0.5
51	8/26/2009 3:10:06 PM	1	0.1	0	0.350000	-0.4045084972	-0.5
52	8/26/2009 3:10:12 PM	1	0.1	0	-0.05	0.15450849723	0.5
53	8/26/2009 3:10:18 PM	1	0.1	0	-0.45	0.15450849718	0.5
54	8/26/2009 3:10:24 PM	1	0.1	0	0.150000	-0.4045084972	-0.5

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 information about configuring database tables with the Cogent DataHub, please refer to [Section 9.5, 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 [Section 10.3, Setting up the DSN \(Data Source Name\)](#)

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 `ODBCTutorial3.g` script uses an SQL **SELECT** command to pull all of the data from a database table.

```
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:

```
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 H. Error Messages

This section presents error numbers that the developer may encounter when using the Cogent DataHub.

H.1. Windows Error Numbers

Number	Error String	Error Description
0	EOK	No error.
1	EPERM	No permissions, or the user is not the process owner.
2	ENOENT	No such file or directory.
3	ESRCH	No such process.
4	EINTR	Interrupted system call.
5	EIO	I/O error.
6	ENXIO	No such device or address.
7	E2BIG	Argument list is too big.
8	ENOEXEC	Executable format is not recognized.
9	EBADF	Bad file number or invalid file descriptor.
10	ECHILD	No child processes exist.
11	EAGAIN	Resource temporarily unavailable or operation would block.
12	ENOMEM	Out of memory.
13	EACCES	Permission denied.
14	EFAULT	Bad memory address.
15	ENOTBLK	Block operation attempted on non-block device.
16	EBUSY	Device or resource busy, or operation already in progress.
17	EEXIST	File exists.
18	EXDEV	Cross-device link.
19	ENODEV	No such device.
20	ENOTDIR	Not a directory.
21	EISDIR	Is a directory.
22	EINVAL	Invalid argument.
23	ENFILE	File table overflow.
24	EMFILE	Too many open files.
25	ENOTTY	Character operation on non-character device.
26	ETXTBSY	Text file is busy.
27	EFBIG	File is too large.
28	ENOSPC	No space left on device.
29	ESPIPE	Illegal seek attempted on a pipe.
30	EROFS	Attempted write to a read-only file system.
31	EMLINK	Too many links.
32	EPIPE	Broken pipe.
33	EDOM	Math argument out of data domain of function.
34	ERANGE	Result too large.

Number	Error String	Error Description
35	EUCLEAN	
36	EDEADLOCK	Deadlock avoided.

H.2. Windows TCP Error Numbers

Number	Error String	Error Description
10035	EWouldBlock	Resource temporarily unavailable or operation would block.
10036	EINPROGRESS	Operation now in progress.
10037	EALREADY	Device or resource busy, or operation already in progress.
10038	ENOTSOCK	Socket operation on non-socket.
10039	EDESTADDRREQ	Destination address required.
10040	EMSGSIZE	Message too long.
10041	EPROTOTYPE	Protocol wrong type for socket.
10042	ENOPROTOOPT	Protocol not available.
10043	EPROTONOSUPPORT	Protocol not supported.
10044	ESOCKTNOSUPPORT	Socket type not supported.
10045	EOPNOTSUPP	Operation not supported.
10046	EPFNOSUPPORT	Protocol family not supported.
10047	EAFNOSUPPORT	Address family not supported by protocol family.
10048	EADDRINUSE	Address already in use.
10049	EADDRNOTAVAIL	Can't assign requested address.
10050	ENETDOWN	Network is down.
10051	ENETUNREACH	Network is unreachable.
10052	ENETRESET	Network dropped connection on reset.
10053	ECONNABORTED	Software caused connection abort.
10054	ECONNRESET	Connection reset by peer.
10054	ECONNRESET	Connection reset by peer.
10055	ENOBUFS	No buffer space available.
10056	EISCONN	Socket is already connected.

Number	Error String	Error Description
10057	ENOTCONN	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.
10058	ESHUTDOWN	Can't send after socket shutdown.
10059	ETOOMANYREFS	Too many references: can't splice.
10060	ETIMEDOUT	Connection timed out.
10061	ECONNREFUSED	Connection refused.
10062	ELOOP	Too many symbolic link or prefix loops.
10063	ENAMETOOLONG	Name too long.
10064	EHOSTDOWN	Host is down.
10065	EHOSTUNREACH	No route to host.
10066	ENOTEMPTY	Directory not empty.
10067	EPROCLIM	Process count limit reached.
10068	EUSERS	User count limit reached.
10069	EDQUOT	Quota limit reached.
10070	ESTALE	Potentially recoverable i/o error.
10071	EREMOTE	Too many levels of remote in path.

H.3. Windows DDE Error Numbers

Number	Error String	Error Description
16384	DMLERR_ADVACKTIMEOUT	Timeout waiting for an advise acknowledge.
16385	DMLERR_BUSY	Recipient is busy.
16386	DMLERR_DATAACKTIMEOUT	Timeout waiting for an advise data acknowledge
16387	DMLERR_DLL_NOT_INITIALIZED	DDEML.DLL is not initialized.
16388	DMLERR_DLL_USAGE	General DDE library usage error.
16389	DMLERR_EXECACKTIMEOUT	Timeout waiting for an exec acknowledgment.
16390	DMLERR_INVALIDPARAMETER	Invalid parameter to DDEML function call.
16391	DMLERR_LOW_MEMORY	Memory is becoming low.
16392	DMLERR_MEMORY_ERROR	Memory is exhausted.

Number	Error String	Error Description
16393	DMLERR_NOTPROCESSED	Receiving task was not interested in message.
16394	DMLERR_NO_CONV_ESTABLISHED	No DDE conversation could be established.
16395	DMLERR_POKEACKTIMEOUT	Timeout waiting for a poke acknowledge.
16396	DMLERR_POSTMSG_FAILED	Attempt to post a window message failed.
16397	DMLERR_REENTRANCY	The DDE library was re-entered during a blocking call.
16398	DMLERR_SERVER_DIED	DDE server has died.
16399	DMLERR_SYS_ERROR	A DDE call has caused a system error.
16400	DMLERR_UNADVACKTIMEOUT	Timeout waiting for an unadvised acknowledge.
16401	DMLERR_UNFOUND_QUEUE_ID	DDE queue id could not be found.

Appendix I. 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 software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, 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 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS 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 J. 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 K. GNU General Public License

GNU General Public License

Version 2, June 1991

Copyright © 1989, 1991 by 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.*

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 L. GNU Lesser General Public License

GNU Lesser General Public License

Version 2.1, February 1999

Copyright © 1991, 1999 by 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.*

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. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

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 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 PERFORMANCE 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.

<one line to give the library’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

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!

I. Cogent DataHub Command Set

Table of Contents

acksuccess	328
add	329
alias	330
alive	331
append	332
assembly	333
asyncsocket	334
attribute	335
auth	336
authgroup	337
authuser	338
auto_create_domains	339
auto_timestamp	340
bridge	341
bridge_remove	343
bridge_remove_pattern	344
bridge_transform	345
cforce	347
cread	348
create	349
create_domain	350
creport	351
cset	352
cwrite	353
DDEAdvise	355
DDEConnect	356
DDEDisconnect	357
DDEInit	358
DDEService	359
DDEUnadvise	360
DDEUnadvisePattern	361
DDEUnadvisePoint	362
debug	363
defaultprop	364
delete	365
deleted	366
div	367
domain	368
domains	369

dump	370
enable_bridging.....	371
enable_dde_client.....	372
enable_dde_server	373
enable_mirror_master.....	374
enable_mirror_slave	375
enable_opc_client.....	376
enable_opc_server.....	377
enable_scripting.....	378
enable_tcp_server	379
error	380
execute_plugin.....	381
exit	382
flush.....	383
flush_log.....	384
force.....	385
format.....	386
heartbeat.....	387
ignore	388
ignore_old_data.....	389
include.....	390
instance	391
load_config_files	392
load_plugin.....	393
load_scripts.....	394
lock	395
log_file	396
log_to_file.....	397
mirror_master	398
mirror_master_2.....	399
mult	401
OPCActivate.....	402
OPCAddItem2.....	403
OPCAppl.....	404
OPCAttach2	405
OPCConnect.....	407
OPCDetach.....	408
OPCEnable.....	409
OPCEnableClient	410
OPCEnableServer.....	411
OPCMinimumSecurity.....	412
OPCModify	413
OPCRefresh.....	415

OPCReload.....	416
OPCRemoveItem	417
private_attribute	418
property	419
quality	420
read.....	421
report	422
report_domain.....	423
report_errors.....	424
request_initial_data	425
save_config.....	426
secure	427
set.....	428
set_canonical	429
show_data	430
show_debug_messages.....	431
show_event_log.....	432
show_icon.....	433
show_properties	434
show_script_log.....	435
subassembly	436
success	437
tcp_service	438
timeout	439
transmit_insignificant.....	440
type	441
unload_plugin.....	442
unreport	443
version	444
write	445

This is the internal command set for the DataHub. For general information on how to use these commands, please refer to [Chapter 22, Using DataHub Commands](#).

acksuccess

acksuccess — tells the DataHub to return success messages.

Syntax

```
(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.

Syntax

```
(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.

Syntax

```
(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 Cascade DataHub that the client is running.

Syntax

(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 Cascade DataHub that it is still up and running.

append

append — appends a string to the value of a point.

Syntax

```
(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.

Syntax

(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 [Section 18.4.2, Assemblies, Subassemblies, Attributes, and Properties](#).

asyncsocket

asyncsocket — sets up asynchronous communication on a socket.

Syntax

(*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*.

attribute

attribute — creates an attribute.

Syntax

(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 [Section 18.4.2, Assemblies, Subassemblies, Attributes, and Properties](#).

auth

auth — requests authentication for a client.

Syntax

```
(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.

authgroup

authgroup — creates a new authentication group.

Syntax

```
(authgroup name bits max_concurrent_logins max_logins expiry)
```

Arguments

name

The name of the group.

bits

Authentication bits.

max_concurrent_logins

The maximum number of concurrent logins for members of this group. Each group member maintains its own independent count.

max_logins

The maximum number of times that a member of this group may log in to the DataHub, ever. Each member of the group maintains its own independent count.

expiry

The expiry date for members of this group. This is given as UNIX epoch time - the number of seconds since Jan. 1, 1970.

Returns

A message indicating success or error. Please refer to [Return Syntax](#) for details.

Description

This command creates a new authentication group.

authuser

authuser — creates a new user.

Syntax

```
(authuser name group hash bits max_concurrent_logins max_logins login_count expiry)
```

Arguments

name

The name of the group.

group

The group name if the user belongs to a group, otherwise " ".

hash

A password hash computed by the DataHub. The algorithm is not published and the hash cannot be reversed to determine the original password.

bits

Authentication bits.

max_concurrent_logins

The maximum number of concurrent logins for members of this group. Each group member maintains its own independent count.

max_logins

The maximum number of times that a member of this group may log in to the DataHub, ever. Each member of the group maintains its own independent count.

login_count

The number of times that this user has logged in, ever.

expiry

The expiry date for members of this group. This is given as UNIX epoch time - the number of seconds since Jan. 1, 1970.

Returns

A message indicating success or error. Please refer to [Return Syntax](#) for details.

Description

This command creates a new user. This cannot really be emitted by a client because a client will not know how to compute the password hash. If a client knows a valid password hash then it can use that hash to produce a new user with the same password as the password that produced that password hash.

auto_create_domains

auto_create_domains — automatically adds domains requested by clients.

Syntax

```
(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.

Syntax

```
(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.

Syntax

```
(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



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 `(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.

Syntax

`(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.

Syntax

(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.

Syntax

```
(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 $(\text{flags} \ \& \ 512) == 0$.

add

The adder value for a linear transformation. This is ignored if $(\text{flags} \ \& \ 512) == 0$.

srcmin

The minimum range map value for the source point. This is ignored if $(\text{flags} \ \& \ 1024) == 0$.

srcmax

The maximum range map value for the source point. This is ignored if $(\text{flags} \ \& \ 1024) == 0$.

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.

Syntax

```
(cforce name value [confidence])
```

Arguments

name

The name of the point. This is a string.

value

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 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.

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 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 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](#) and [force](#).

cread

cread — creates and reads a point.

Syntax

`(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.

Syntax

```
(create name [0|1])
```

Arguments

name

The name of the point, as a string.

0 | 1

Tells **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 **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 [cset](#).

Example

Using the Gamma `datahub_command` function, you could pass a `create` command to the DataHub as follows to create `MyNewPoint` in the `default` domain, and assign it a value of 1.

```
datahub_command ("(create default:MyNewPoint)", 1);
```

create_domain

create_domain — creates a new domain.

Syntax

`(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.

Syntax

(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.

Syntax

```
(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. 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.

Syntax

```
(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.

Syntax

```
(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 F, DDE Overview](#).

DDEConnect

DDEConnect — makes a connection to a DDE service and topic.

Syntax

```
(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 disconnect 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 F, DDE Overview](#).

DDEDisconnect

DDEDisconnect — disconnects and discards a DDE connection.

Syntax

(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.

DDEInit

DDEInit — initializes the DataHub to act as a DDE server.

Syntax

(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

DDEService — assigns a DDE service name.

Syntax

(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 Cascade 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.

Syntax

`(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

DDEUnadvisePattern — removes multiple items from a DDE connection.

Syntax

(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.

Syntax

(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

debug — sets the debug level.

Syntax

`(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.

Syntax

(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 [Section 18.4.2, Assemblies, Subassemblies, Attributes, and Properties](#).

delete

delete — deletes a point—use with caution.

Syntax

```
(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.

deleted

deleted — checks if a point has been deleted.

Syntax

```
(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.

div

div — does division on the value of a point.

Syntax

(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.

Syntax

(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 Cascade DataHub the domain that the client is using.

domains

domains — lists all domains in the DataHub.

Syntax

(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.

Syntax

`(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.

enable_bridging

enable_bridging — enables or disables bridging capabilities

Syntax

(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 Cascade DataHub to support [bridging](#).

See Also

[bridge](#)

enable_dde_client

enable_dde_client — enables or disables DDE client capabilities.

Syntax

(enable_dde_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 Cascade 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.

Syntax

```
(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 Cascade 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_mirror_master

enable_mirror_master — enables or disables mirror master capabilities.

Syntax

```
(enable_mirror_master 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 Cascade DataHub to act as a mirror master. This corresponds to the **Act as a master...** checkbox in the [Tunnel/Mirror](#) option of the Properties window. For more information about tunnelling/mirroring, please refer to [Section 20.4, Tunnel/Mirror](#).

enable_mirror_slave

enable_mirror_slave — enables or disables mirror slave capabilities.

Syntax

```
(enable_mirror_slave 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 Cascade DataHub to act as a mirror slave. This corresponds to the Act as a tunnelling/mirroring slave... checkbox in the [Tunnel/Mirror](#) option of the Properties window. For more information about tunnelling/mirroring, please refer to [Section 20.4, Tunnel/Mirror](#).

enable_opc_client

enable_opc_client — enables or disables OPC client capabilities.

Syntax

(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 Cascade 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.

Syntax

(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 Cascade 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

Syntax

```
(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 Cascade DataHub to support [scripting](#).

enable_tcp_server

enable_tcp_server — enables or disables TCP server capabilities.

Syntax

```
(enable_tcp_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 Cascade DataHub to act as a TCP server.

error

error — sends an error with an error string.

Syntax

`(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*)

Syntax

(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.

Syntax

```
(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.

flush

flush — flushes output to a terminal (Linux).

Syntax

(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 — forces an immediate update of the Script Log (Windows).

Syntax

(flush_log)

Arguments

none

Returns

A message indicating success or error. Please refer to [Return Syntax](#) for details.

Description

This command forces an immediate update of the Script Log. This is rarely necessary.

force

force — forces a write to a point.

Syntax

```
(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 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 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 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](#) and [cforce](#).

format

format — is an efficiency enhancement for Linux.

Syntax

(*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.

heartbeat

heartbeat — establishes a heartbeat message.

Syntax

(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.

ignore

ignore — ignores a given point.

Syntax

`(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.

Syntax

```
(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.

Syntax

```
(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.

Syntax

```
(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 [Section 18.4, Data Organization](#).

load_config_files

load_config_files — loads configuration files.

Syntax

```
(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*)

Syntax

```
(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.

Syntax

```
(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.

Syntax

```
(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.

Syntax

`(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_to_file

log_to_file — starts or stops logging to a file.

Syntax

```
(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 tunnelling (mirroring) master.

Syntax

```
(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 the entries in the Tunnel Master dialog that opens when you click the **Add** button in the [Tunnel/Mirror](#) option of the Properties window.



The DataHub uses mirroring to do tunnelling, so the term 'mirror' is used to name the command.

mirror_master_2

mirror_master_2 — sets up a secure tunnelling (mirroring) master.

Syntax

```
(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:

Hex	Flag	Notes
0x00100000	READABLE	Data is readable from the master. Always set this.
0x00200000	WRITABLE	Data is writable to the master.
0x01000000	SYNC_SEND	On initial connection, send all data to the master
0x02000000	SYNC_RECV	On initial connection, receive all data from the master
0x04000000	SYNC_TIME	On initial connection, synchronize by comparing time stamps.
0x10000000	AUTHORITATIVE	Master is authoritative. Data here is marked Not Connected when the connection drops.
0x20000000	NON_AUTHORITATIVE	I am authoritative. Data in the master is marked Not Connected when the connection drops.
0x40000000	OVERRIDE_TIME	Override the time stamp on incoming data with the system clock time.

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 Tunnel Master dialog that opens when you click the Add button in the [Tunnel/Nirror](#) option of the Properties window.



The DataHub uses mirroring to do tunnelling, so the term 'mirror' is used to name the command.

mult

mult — multiplies the value of a point.

Syntax

```
(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.

Syntax

```
(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.

Syntax

```
(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

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 adds OPC items to an OPC server connection. It does not take effect until the [OPCApply](#) command is issued.

Example

```
(OPCAddItem2 "MyOPCServer" 2 0 "Tank3.Level" ("Tank3" "Level"))
```

OPCApply

OPCApply — applies changes to an outgoing connection.

Syntax

```
(OPCApply [label])XF
```

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 [OPCAddItem2](#) and [OPCRemoveItem](#). Therefore, you can call **OPCAddItem2** and **OPCRemoveItem** multiple times and then must issue **OPCApply** once for the changes to take effect.

OPCAttach2

OPCAttach2 — sets up an OPC connection.

Syntax

```
(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.

Syntax

```
(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.

Syntax

(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

OPCEnable — enables or disables outgoing OPC connections.

Syntax

```
(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.

Syntax

(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.

Syntax

(OPCEnableServer 0|1)

Arguments

0|1

If 1, the 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 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.

Syntax

(OPCMinimumSecurity 0|1)

Arguments

0 | 1

If 1, the 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 DataHub attempts to override the current system COM security settings when it starts. If you change this setting, you must re-start the 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.

Syntax

```
(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 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.

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.

OPCRefresh

OPCRefresh — sends a **Refresh2** command to the OPC server.

Syntax

```
(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.

Syntax

```
(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.

OPCRemoveItem

OPCRemoveItem — removes an item based on its DataHub point name.

Syntax

```
(OPCRemoveItem 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 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 DataHub point name. OPC items are mapped in a one-to-many relationship with 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.

Syntax

```
(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 [Section 18.4.3, Attributes and Types](#).

property

property — creates a property for an assembly.

Syntax

```
(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 [Section 18.4.2, Assemblies, Subassemblies, Attributes, and Properties](#).

quality

quality — assigns a quality to a point.

Syntax

(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.

Syntax

`(read name)`

Arguments

name

The name of a 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 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.

Syntax

`(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

report_domain — registers points and requests information on a whole domain.

Syntax

(report_domain domain flags)

Arguments

domain

The desired domain.

flags

Any bitwise-or combination of:

DH_REG_ALL	Register all points on this domain.
DH_REG_FUTURE	Register any new points created on this domain subsequent to this call.
DH_REG_QUALIFY	Tell the Cascade DataHub to emit all point names with the domain prepended, as in <i>domain:point_name</i> .
DH_REG_ONCEONLY	Tell the Cascade DataHub to send each point value exactly once.
DH_REG_MODEL	Tell the Cascade DataHub to send the data model as well as the point values.

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 be 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.

Syntax

```
(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 Cascade DataHub.

request_initial_data

request_initial_data — gets current data when client connection is made.

Syntax

(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 Cascade DataHub to send the current value of all points when the client connects.

save_config

save_config — forces the DataHub to save its configuration.

Syntax

(save_config)

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.

Syntax

```
(secure 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.

Syntax

```
(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_canonical

set_canonical — sets the type of a point.

Syntax

```
(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.

Syntax

(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.

Syntax

(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.

Syntax

(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

show_icon — displays the system tray icon.

Syntax

(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 try icon ().

show_properties

show_properties — displays the Properties window.

Syntax

```
(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.

Syntax

```
(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.

Syntax

(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 [Section 18.4.2, Assemblies, Subassemblies, Attributes, and Properties](#).

success

success — sets up a success message.

Syntax

(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.

Syntax

```
(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.

Syntax

(**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 Cascade DataHub to stop sending data to the client for *ms* milliseconds.

transmit_insignificant

transmit_insignificant — permits transmission of insignificant changes.

Syntax

```
(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.

type

type — creates a type.

Syntax

```
(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 [Section 18.4.3, Attributes and Types](#).

unload_plugin

unload_plugin — unloads a plugin. (*experimental*)

Syntax

(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.

Syntax

`(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.

version

version — returns the current version number.

Syntax

(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.

Syntax

```
(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_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. See also [cwrite](#).

II. Obsolete and Unused Commands

Table of Contents

bandwidth_reduce	448
drop_license	449
echo	450
enable_connect_server	451
EnableDDEServer	452
exception_buffer	453
failed_license	454
master_host	455
master_service	456
on_change	457
OPCAddItem	458
OPCAttach	459
OPCInit	460
point	461
qnx_name_attach	462
qnx_receiver	463
readid	464
register_datahub	465
report_all	466
report_datahubs	467
request	468
run	469
script_register	470
script_symbol	471
slave	472
sync	473
taskdied	474
taskstarted	475
using_license	476
warn_of_license_expiry	477

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.

bandwidth_reduce

`bandwidth_reduce` — is for internal use only.

drop_license

`drop_license` — is for internal use only.

echo

echo — is for internal use.

Syntax

(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.

Syntax

```
(enable_connect_server 0|1)
```

EnableDDEServer

EnableDDEServer — is for internal use only.

Syntax

```
(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.

Syntax

`(exception_buffer bytes)`

failed_license

failed_license — is for internal use only.

master_host

master_host — is deprecated in favor of **mirror_master**.

Syntax

(master_host name|IP)

master_service

master_service — is deprecated in favor of **mirror_master**.

Syntax

```
(master_service service|port)
```

on_change

on_change — is for internal use only.

OPCAddItem

OPCAddItem — is deprecated in favor of **OPCAddItem2**.

Syntax

```
(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**.

Syntax

```
(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 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.

Syntax

`(OPCInit)`

point

point — is used internally.

Syntax

```
(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.

Syntax

`(qnx_name_attach node name)`

qnx_receiver

`qnx_receiver` — does nothing.

Syntax

`(qnx_receiver name)`

readid

readid — should *not* be used.

Syntax

(readid pointnumber)

Arguments

pointnumber

The n^{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^{th} point in the point table of the sender's `default` domain. This is neither useful nor robust, since deleting a point will cause unpredictable behavior. Avoid using this command.

register_datahub

`register_datahub` — replaced by [report_domain](#).

Syntax

```
(register_datahub domain)
```

report_all

`report_all` — replaced by [report_domain](#).

Syntax

```
(report_all future domain_needed)
```


report_datahubs

`report_datahubs` — does nothing.

Syntax

```
(report_datahubs yes|no)
```

request

request — replaced by [report_domain](#).

Syntax

(request *pointname*)

run

run — does nothing.

Syntax

`(run command [argument...])`

script_register

`script_register` — is for internal use only.

script_symbol

`script_symbol` — is for internal use only.

slave

slave — is for internal use only.

sync

sync — is for internal use only.

taskdied

taskdied — is for internal use.

Syntax

(taskdied *name down qu nodename node pid chid*)

taskstarted

taskstarted — is for internal use.

Syntax

(taskstarted *name domn qu nodename node pid chid*)

using_license

`using_license` — is for internal use only.

warn_of_license_expiry

`warn_of_license_expiry` — is deprecated.

Syntax

```
(warn_of_license_expiry 0|1)
```

Index

A

- acksuccess, [328](#)
- activation
 - for bridging, [228](#)
- add, [329](#)
 - configuration file, [257](#)
 - data domain, [213](#)
 - DDE client, [232](#)
 - DDE server, [233](#)
 - Security client, [260](#)
- add a timestamp, [212](#)
- Aggregation
 - and Bridging, [51](#)
 - and Tunnelling, [51](#)
 - configuring the DataHub, [44](#)
 - OPC, [44](#)
- alias, [330](#)
- alive, [331](#)
- append, [332](#)
- assembly, [333](#)
- asyncsocket, [334](#)
- attribute, [335](#)
- auth, [336](#)
- authentication, [188](#)
- authgroup, [337](#)
- authorization, [190](#)
- authuser, [338](#)
- auto_create_domains, [339](#)
- auto_timestamp, [340](#)

B

- bandwidth_reduce, [448](#)
- bridge, [341](#)
- Bridges
 - configuring, [53](#)
- bridge_remove, [343](#)
- bridge_remove_pattern, [344](#)
- bridge_transform, [345](#)
- bridging
 - activation, [228](#)
 - configuration window, [228](#)
 - creating data sets, [59](#)
 - direction, [231](#)
 - local servers, [58](#)
 - OPC, [53](#)
 - point display, [231](#)
 - point selection, [229](#)

C

- status, [228](#)
- to Excel, [60](#)
- transformations, [230](#)
- Bridging Properties, [227](#)
- cforce, [347](#)
- client
 - OPC, [214](#)
 - OPC A&E, [223](#)
- Cogent DataHub
 - exiting, [13](#)
 - installing, [1](#)
 - running, [13](#)
 - uninstalling, [1](#)
- COM
 - security, [220](#)
- Computer Name
 - OPC, [214](#)
 - OPC A&E, [221](#)
- configuration file
 - add, remove, move, load, [257](#)
- configuration window
 - bridging, [228](#)
- configure
 - database, [240](#)
 - email, [250](#)
 - historian, [244](#)
 - quicktrend, [233](#)
 - redundancy, [251](#)
 - System Monitor, [247](#)
 - web server, [238](#)
 - webview, [237](#)
- configure database
 - for data logging, [240](#)
- Configured actions
 - database queries, [140](#)
 - email, [161](#)
 - logging, [125](#)
- Configuring
 - bridges, [53](#)
 - the mail server, [149](#)
- Configuring a database table, [113](#)
 - for database queries, [131](#)
- Configuring the DataHub
 - for aggregation, [44](#)
- connecting
 - to InTouch, [178](#)
 - to Linux, [166](#)
 - to QNX, [172](#)
- Connection Name

- DDE, [232](#)
- OPC, [214](#)
- OPC A&E, [221](#)
- connections
 - Excel and Cogent DataHub, [61](#)
- CPU saturation
 - optimizing, [208](#)
- cread, [348](#)
- create, [349](#)
- create_domain, [350](#)
- creport, [351](#)
- cset, [352](#)
- cwrite, [353](#)

D

- data domain
 - add, [213](#)
 - remove, [213](#)
- Data Domain Name
 - OPC, [214](#)
 - OPC A&E, [221](#)
- data logging
 - configure database, [240](#)
 - how to use, [105](#)
 - store and forward, [241](#)
- data sets
 - creating to, [59](#)
- data throughput
 - optimizing, [206](#)
- database
 - concepts, [302](#)
 - configure, [240](#)
 - query, [243](#)
 - terminology, [302](#)
 - write to, [240](#)
- Database Properties, [240](#)
- Database queries
 - assigning a trigger, [134](#)
 - configured actions, [140](#)
 - configuring a database table, [131](#)
 - how to, [126](#)
 - quick start, [126](#)
 - setting up DSN, [129](#)
 - trigger conditions, [136](#)
- DataHub
 - scripting;, [183](#)
 - security;, [186](#)
- DDE
 - non-English characters, [233](#)
- DDE client
 - add, remove, edit, enable, [232](#)

- DDE Item Definition Window, [232](#)
- DDE Properties, [231](#)
- DDE server
 - add, remove, enable, [233](#)
- DDEAdvise, [355](#)
 - with drag and drop, [61](#)
- DDEConnect, [356](#)
- DDEDisconnect, [357](#)
- DDEInit, [358](#)
- DDERequest
 - in a macro, [63](#)
- DDEService, [359](#)
- DDEUnadvise, [360](#)
- DDEUnadvisePattern, [361](#)
- DDEUnadvisePoint, [362](#)
- debug, [363](#)
- defaultprop, [364](#)
- delete, [365](#)
- deleted, [366](#)
- direction
 - for bridging, [231](#)
- div, [367](#)
- domain, [368](#)
- domains, [369](#)
- drop_license, [449](#)
- DSN
 - setting up for logging, [111](#)
 - setting up for queries, [129](#)
- dump, [370](#)

E

- echo, [450](#)
- edit
 - DDE client, [232](#)
 - Security client, [260](#)
- Email
 - assigning a trigger, [156](#)
 - configure, [250](#)
 - configured actions, [161](#)
 - defining a message, [154](#)
 - how it works, [148](#)
 - how to send, [148](#)
 - HTML message examples, [161](#)
 - test message, [150](#)
 - trigger conditions, [158](#)
- Email Properties, [250](#)
- Email server
 - configuring, [149](#)
- EnableDDEServer, [452](#)
- enable_bridging, [371](#)
- enable_connect_server, [451](#)

- enable_dde_client, [372](#)
- enable_dde_server, [373](#)
- enable_mirror_master, [374](#)
- enable_mirror_slave, [375](#)
- enable_opc_client, [376](#)
- enable_opc_server, [377](#)
- enable_scripting, [378](#)
- enable_tcp_server, [379](#)
- enter
 - licenses, [264](#)
- error, [380](#)
- Excel
 - arrays, [74](#)
 - bridging to, [60](#)
 - connecting to the Cogent DataHub, [61](#)
 - data from, [64](#)
 - data into, [61](#)
 - data to DDE Client, [65](#)
 - example, [69](#)
 - macro to get data, [63](#)
 - macro to send data, [67](#)
 - networking, [71](#)
 - ranges, [74](#)
- exception_buffer, [453](#)
- execute_plugin, [381](#)
- exit, [382](#)
- exiting the DataHub, [13](#)

F

- failed_license, [454](#)
- flush, [383](#)
- flush_log, [384](#)
- force, [385](#)
- format, [386](#)

G

- General Properties, [212](#)

H

- heartbeat, [387](#)
- help
 - pop-up, [13](#)
- historian
 - configure, [244](#)
- Historian Properties, [244](#)
- How to
 - query a database, [126](#)
- How to send
 - email, [148](#)

- text messages, [148](#)

How to use

- data logging, [105](#)
- the mailer, [148](#)
- the System Monitor, [141](#)
- the web server, [84](#)

HTML message examples

- email, [161](#)

I

- ignore, [388](#)
- ignore_old_data, [389](#)
- include, [390](#)
- installing
 - updates, [7](#)
- installing the Cogent DataHub, [1](#)
- instance, [391](#)
- InTouch
 - connections, [178](#)
- Item Names
 - DDE, [232](#)

K

Key columns

- for logging, [117](#)

L

- licenses
 - entering, [264](#)
 - loading, [263](#)
 - removing, [264](#)
 - string, [263](#)
- Licenses Properties, [263](#)
- Linux
 - connections, [166](#)
 - to OPC, Linux setup, [170](#)
 - to OPC, Windows setup, [166](#)
- load
 - configuration file, [257](#)
- load_config_files, [392](#)
- load_plugin, [393](#)
- load_scripts, [394](#)
- lock, [395](#)
- Logging
 - assigning a trigger, [118](#)
 - configured actions, [125](#)
 - configuring a database table, [113](#)
 - key columns, [117](#)
 - quick start, [105](#)

- setting up DSN, [111](#)
- trigger conditions, [120](#)
- log_file, [396](#)
- log_to_file, [397](#)

M

- macro to get data, [63](#)
- macro to send data, [67](#)
- Mail server
 - configuring, [149](#)
- Mailer
 - how to use, [148](#)
- master
 - for tunnelling/mirroring, [227](#)
- master_host, [455](#)
- master_service, [456](#)
- message buffer
 - maximum size, [212](#)
- mirror_master, [398](#)
- mirror_master_2, [399](#)
- move
 - configuration file, [257](#)
- mult, [401](#)

N

- networking
 - Excel, [71](#)
 - System Monitor, [144](#)

O

- ODBC, [302](#)
- old value queuing
 - optimizing, [206](#)
- on_change, [457](#)
- OPC
 - Aggregation, [44](#)
 - Bridging, [53](#)
 - client, [214](#)
 - Computer Name, [214](#)
 - Connection Name, [214](#)
 - Data Domain Name, [214](#)
 - server, [220](#)
 - Server Name, [214](#)
 - to Linux, Linux setup, [170](#)
 - to Linux, Windows setup, [166](#)
 - to QNX, QNX setup, [176](#)
 - to QNX, Windows setup, [172](#)
 - Tunnelling, [29](#)
- OPC A&E

- client, [221](#)
- Computer Name, [221](#)
- Connection Name, [221](#)
- Data Domain Name, [221](#)
- server, [221](#)
- Server Name, [221](#)
- OPC DA Properties, [214](#)
- OPC Properties, [221](#)
- OPCActivate, [402](#)
- OPCAddItem, [458](#)
- OPCAddItem2, [403](#)
- OPCApply, [404](#)
- OPCAttach, [459](#)
- OPCAttach2, [405](#)
- OPCConnect, [407](#)
- OPCDetach, [408](#)
- OPCEnable, [409](#)
- OPCEnableClient, [410](#)
- OPCEnableServer, [411](#)
- OPCInit, [460](#)
- OPCMinimumSecurity, [412](#)
- OPCModify, [413](#)
- OPCRefresh, [415](#)
- OPCReload, [416](#)
- OPCRemoveItem, [417](#)
- optimizing
 - CPU saturation, [208](#)
 - data throughput, [206](#)
 - old value queuing, [206](#)
 - screen output, [207](#)
 - slow networks, [206](#)
 - TCP connections, [206](#)
 - un-buffered delivery, [207](#)
- options
 - Web Server, [239](#)

P

- passwords, [196](#)
- point, [461](#)
- point display
 - for bridging, [231](#)
- point selection
 - for bridging, [229](#)
- pop-up
 - help, [13](#)
- private_attribute, [418](#)
- Properties
 - Bridging, [227](#)
 - Database, [240](#)
 - DDE, [231](#)
 - Email, [250](#)

- General, [212](#)
- Historian, [244](#)
- Licenses, [263](#)
- OPC A&E, [221](#)
- OPC DA, [214](#)
- Quick Trend, [233](#)
- Redundancy, [251](#)
- Scripting, [257](#)
- Security, [260](#)
- System Monitor, [247](#)
- Tunnelling/Mirroring, [223](#)
- Web Server, [238](#)
- WebView, [237](#)
- property, [419](#)

Q

- QNX
 - connections, [172](#)
 - to OPC, QNX setup, [176](#)
 - to OPC, Windows setup, [172](#)
- qnx_name_attach, [462](#)
- qnx_receiver, [463](#)
- quality, [420](#)
- query
 - database, [243](#)
- Quick start
 - database queries, [126](#)
 - logging, [105](#)
- quicktrend
 - configure, [233](#)
- QuickTrend Properties, [233](#)

R

- read, [421](#)
- readid, [464](#)
- reading data, [197](#)
- redundancy
 - configure, [251](#)
- Redundancy Properties, [251](#)
- register_datahub, [465](#)
- remove
 - configuration file, [257](#)
 - data domain, [213](#)
 - DDE client, [232](#)
 - DDE server, [233](#)
 - licenses, [264](#)
 - Security client, [260](#)
- report, [422](#)
- report_all, [466](#)
- report_datahubs, [467](#)

- report_domain, [423](#)
- report_errors, [424](#)
- request, [468](#)
- request_initial_data, [425](#)
- run, [469](#)
- running the DataHub, [13](#)

S

- save_config, [426](#)
- screen output
 - optimizing, [207](#)
- scripting
 - DataHub, [183](#)
- Scripting Properties, [257](#)
- script_register, [470](#)
- script_symbol, [471](#)
- secure, [427](#)
- security
 - authentication, [188](#)
 - authorization, [190](#)
 - COM, [220](#)
 - DataHub, [186](#)
 - passwords, [196](#)
- Security client
 - add, remove, edit, enable, [260](#)
- Security Properties, [260](#)
- server
 - OPC, [220](#)
 - OPC A&E, [223](#)
- Server Name
 - OPC, [214](#)
 - OPC A&E, [221](#)
- Service
 - DDE, [232](#)
- set, [428](#)
- set_canonical, [429](#)
- show_data, [430](#)
- show_debug_messages, [431](#)
- show_event_log, [432](#)
- show_icon, [433](#)
- show_properties, [434](#)
- show_script_log, [435](#)
- slave, [472](#)
 - for tunnelling/mirroring, [224](#)
- slow networks
 - optimizing, [206](#)
- SMS text messaging, [148](#)
- SQL, [303](#)
- status
 - of bridging, [228](#)
- store

- transformations, [229](#)
- store and forward
 - for data logging, [241](#)
- string
 - license, [263](#)
- subassembly, [436](#)
- success, [437](#)
- sync, [473](#)
- System Monitor
 - configure, [247](#)
 - how to use, [141](#)
 - networking, [144](#)
- System Monitor Properties, [247](#)
- system requirements, [1](#)

T

- taskdied, [474](#)
- taskstarted, [475](#)
- TCP connections
 - optimizing, [206](#)
- tcp_service, [438](#)
- Text messages
 - how to send, [148](#)
- Text messaging
 - with SMS, [148](#)
- timeout, [439](#)
- Topic
 - DDE, [232](#)
- transformations
 - for bridging, [230](#)
 - storing, [229](#)
- transmit changes
 - insignificant, [212](#)
 - old timestamp, [212](#)
- transmit_insignificant, [440](#)
- Trigger
 - for database queries, [134](#)
 - for emails, [156](#)
 - for logging, [118](#)
- Trigger conditions
 - database queries, [136](#)
 - email, [158](#)
 - logging, [120](#)
- Tunnelling
 - and Bridging, [41](#)
 - and Aggregation, [42](#)
 - connecting to an OPC client, [37](#)
 - connecting to an OPC server, [30](#)
 - OPC, [29](#)
 - part of the data set, [41](#)
 - testing the connection, [41](#)

- tunnelling/mirroring
 - master, [227](#)
 - slave, [224](#)
- Tunnelling/Mirroring Properties, [223](#)
- type, [441](#)

U

- un-buffered delivery
 - optimizing, [207](#)
- uninstalling the DataHub, [1](#)
- unload_plugin, [442](#)
- unreport, [443](#)
- updates
 - installing, [7](#)
- using_license, [476](#)

V

- version, [444](#)

W

- warn_of_license_expiry, [477](#)
- web server
 - configure, [238](#)
 - how to use, [84](#)
 - options, [239](#)
- Web Server Properties, [238](#)
- webview
 - configure, [237](#)
- WebView Properties, [237](#)
- write, [445](#)
- write to
 - database, [240](#)
- writing data, [197](#)

Colophon

This book was produced by Cogent Real-Time Systems, Inc. from a single-source group of SGML files. Gnu Emacs was used to edit the SGML files. The DocBook DTD and related DSSSL stylesheets were used to transform the SGML source into HTML, PDF, and QNX Helpviewer output formats. This processing was accomplished with the help of OpenJade, JadeTeX, Tex, and various scripts and makefiles. Details of the process are described in our book: *Preparing Cogent Documentation*, which is published on-line at

<http://developers.cogentrts.com/cogent/prepdoc/book1.html>.

Text and illustrations by Bob McIlvride and Paul Benford.