Documentation Library

DataHub® APIs for C++, Java, and .NET

Version 7.2

Cogent Real-Time Systems, Inc.

January 13, 2011

DataHub® APIs for C++, Java, and .NET: Version 7.2

The C++, Java, and .NET Application Program Interfaces for the DataHub.

Published January 13, 2011
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

Revision History

Revision 7.2-1 September 2007
Updated method and Java applet reference.
Revision 6.2-1 April 2005
Initial release of documentation.

Copyright, trademark, and software license information.

Copyright Notice

Trademark Notice

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

Chapter 1. Introduction

1.1. When to use the different Cogent APIs

There are four Cogent APIs, grouped as:

- DataHub APIs for C++, Java, and .NET
- Cogent C API

The DataHub APIs for C++, Java, and .NET

These three APIs share, as much as possible, common methods and syntax. For this reason they are distributed in one package and documented in a single book.

- **The DataHub API for C++** lets you write programs in C++ that connect to the DataHub over TCP, namely LAN, WAN, or the Internet.
- **The DataHub API for Java** lets you write programs in Java that connect to the DataHub over TCP, namely LAN, WAN, or the Internet. In addition, it lets you create web browser applications that receive and display live data from the DataHub.
- **The DataHub API for .NET** lets you write programs in .NET that connect to the DataHub over TCP, namely LAN, WAN, or the Internet. This API is implemented in C#, but can be used with any .NET language.

The following table shows the availability and support for these APIs in Windows, Linux, and QNX:

Language	Windows	Linux	QNX 6	QNX 4
C++	supported	unsupported	unsupported	unsupported
Java	supported	unsupported	unsupported	not available
.NET	supported	may be available using Mono	not available	not available

The Cogent C API

This API lets you write high-speed clients that can interact with the Cascade DataHub, Cascade Historian, Cascade TextLogger, CIF Driver, DVN Driver, PFB Driver, and Gamma. It works in Linux, QNX 6, and QNX 4. Interprocess communication relies on Send/Receive/Reply message passing. In Linux, this is supported by Cogent's SRR Module. In QNX, this is supported by QNX's own Send/Receive/Reply protocol.

For more information on this API, please refer to the Cogent C API manual.

1.2. Preliminaries

The DataHub APIs for C++, Java, and .NET are made up of two classes, `DataHubConnector` and `DataHubPoint` whose methods allow you to interface with the DataHub.

System Requirements

The DataHub APIs for C++, Java, and .NET are compatible with:

- Windows XP Home & Professional
- Windows 2000
- Windows NT 4.0 - All Service Packs should be installed.

Installation

To install the DataHub APIs for C++, Java, and .NET from an archive downloaded from the Cogent web site, follow these steps:

1. Double-click on the program archive `DataHubAPI6.4-xxxxxx-Windows.exe`.
2. Follow the instructions.

To install the DataHub APIs for C++, Java, and .NET from the Cascade Middleware CD, follow these steps:

1. Insert the CD into the drive.
2. Follow the instructions.

1.3. C++ Programming

The C++ API is intended for application programmers who are working in an unmanaged C++ environment in Windows MFC, Windows ATL, Linux, or QNX.

To optimize throughput between your program and the DataHub when using the C++ API, you can use the **sendBinaryPointMessages** method.

Include Statement for Windows:

```
#include <DataHubConnector.h>
```

Include Statement for Linux, QNX4, and QNX6:

```
#include <cogent/DataHubConnector.h>
```

1.4. Java Programming

The Java API implements the `DataHubConnector` class as the basic class used to communicate with the DataHub. Programmers writing stand-alone applications need only the `DataHubConnector` and `DataHubPoint` classes.

Java Class Overview

The following classes are included in the Java API installation. They are informally arranged here to give some idea of the interrelationships:

Classes used for general programming

- **`DataHubConnector`** provides connectivity to the DataHub.
  - **`DataHubEventConsumer`** implements callbacks for `DataHubConnector`.
    - **`DataHubEventDispatcher`** is an interface that extends `DataHubEventConsumer` class, and is used by the `DataHubBaseApplet` class (see below).

- **`DataHubPoint`** represents DataHub point objects.

    Classes used for web programming

- **`DataHubBaseApplet`** is the applet that makes connections to the DataHub. It provides access to all the data in a single domain. There need be only one `DataHubBaseApplet` per HTML page, because indivual connections are made using `DataHubListener` (see below). The following two classes extend the `DataHubBaseApplet` class:
    - **`DataHubViewer`** displays a table of all the data in the domain. It implements `DataHubEventConsumer`.
    - **`DataHubLink`** is used to instantiate a `DataHubBaseApplet` for supporting `DataHubListener` widgets. It embeds a small text message "Powered by Cogent" in the page, and it's color: red, yellow, or green, indicates the status of the link.

- **`DataHubListener`** is an applet that gets data from a specific point or points in the DataHub. It is a parasite in the sense that it relies on the connection to the DataHub provided by a `DataHubBaseApplet`. You can use any number of `DataHubListener`s per HTML page, without noticably affecting the rate of data throughput to the page. The following three widgets are extended from `DataHubListener`:
    - **`DataHubLabel`** displays the value of a DataHub point.
    - **`DataHubEntryField`** is an entry field for changing the value of a DataHub point.
    - **`DataHubButton`** sends a value for a DataHub point.
    - **`DataHubToggleButton`** toggles a DataHub point between two values.
    - **`DataHubCheckBox`** toggles a DataHub point between two values.
    - **`DataHubRadioButton`** a special button, used in a `DataHubRadioGroup`.
    - **`DataHubRadioGroup`** a group of `DataHubRadioButton`s that provides a way to select one of several mutually-exclusive values for a DataHub point.
    - **`DataHubSlider`** changes the value of a DataHub point by sliding a pointer.
    - **`DataHubSpinner`** sets the value of a DataHub point using up and down arrows.
    - **`DataHubProgressBar`** gives a graphical representation of the value of a DataHub point.

- **`DataHubDummy`** is provided as a convenience to the HTML programmer.

    For internal use

- **`DataHubRendezvous`** provides a meeting point for the `DataHubBaseApplet` and all of the `DataHubListener` widgets on the page. It is for internal use, providing static data that gets initialized before any applet starts, giving all the applets a means of finding one another.

Import Statements

```
import cogent.*;
```

or

```
import cogent.DataHubConnector;
import cogent.DataHubPoint;
```

1.5. .NET Programming

The .NET API is written in C#, and implements the `DataHubConnector` class. You can compile the file `DHNetAPI.cs` to create a .NET library that can be used by any .NET language, such as Visual Basic .NET.

The .NET installation includes a test program that can be used to connect to an existing DataHub to view data graphically. The two data sets supported are the "DataSim" data generated by the local DataSim program, or the "test" data generated by the Internet data set at http://developers.cogentrts.com. The .NET test program will arrange its graph based on the domain name chosen.

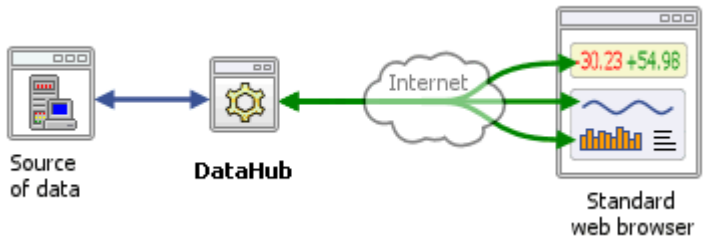
Requirement Statement

```
using Cogent.DataHubAPI;
```

Chapter 2. Web Programming with Java

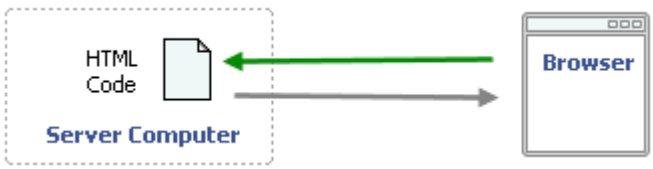
2.1. Overview

The DataHub lets you put live data into Java applets that are embedded in a web page. When the someone visits the web page, the Java applet attaches to the DataHub and the applet starts to receive live data updates without refreshing the page.

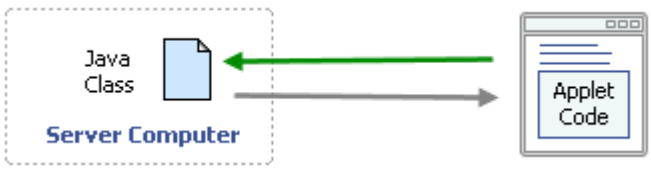


Here is an overview of how your code gets live data from the DataHub to a web page.

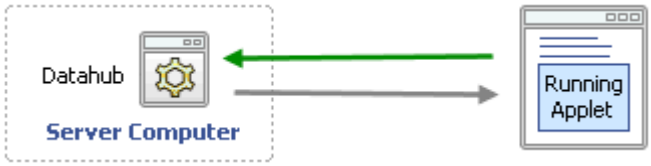
1. The Browser Loads the HTML
    a. The web browser follows the URL to the web server.
    b. The web server returns the HTML code for the page.



2. The Browser Loads the Java code
    a. The browser reads the HTML code and encounters the embedded APPLET code.
    b. The APPLET code instructs the browser to go back to the web server and load the Java code.
    c. The web server provides the Java code.



3. The Browser runs the Java code
    a. The browser runs the Java code (in the Java Virtual Machine).
    b. The Java code instructs the Java VM to connect to the DataHub running on the server computer. From that point on, all communication is between the Java applet and the DataHub only. The web server is not involved.
    c. In the browser, the Java code tells the Java plug-in how to display the information coming from the DataHub.



2.2. Working with Java Applets

The DataHub offers a mechanism for displaying live data in a web browser through Sun's Java plug-in. The DataHub API defines a number of Java applets, each of which can display one or more data points. An applet can be as simple as a text field, or as complex as a data table. Web pages typically contain more than one applet, each displaying a specific item of information.

2.2.1. Connecting

Java applets connect to the DataHub via a private TCP connection. This connection uses the the Tunnel/Mirror functionality of the DataHub. So you must configure your DataHub as a Tunnelling/Mirror Master for a Java applet to connect. The Java applet must be configured to use the same port number that the DataHub is configured to use in the mirroring or tunnelling master configuration.

There are two types of applets supplied with the DataHub API. The first is a `DataHubBaseApplet`, which is an applet that opens the TCP socket to the DataHub and manages the data flow between the browser and the DataHub. The second type is a `DataHubListener`, which is a screen element that communicates with a `DataHubBaseApplet` for its live data. This arrangement means that even though there may be several applets on a single web page, they can share a single connection to the DataHub. This is significant since the DataHub requires a TCP Link License for each concurrent connection.

2.2.2. Applet Configuration

Java applets are configured through parameters in the HTML source for the web page. Different applets can accept different arguments. No argument may appear more than once per applet. Please refer to Section 2.5.2, *Example HTML coding* for examples.

2.2.3. Specifying Colors

Some applet parameters require a color specification. Colors are specified in hexadecimal as RGB values. Colors may optionally contain an alpha blending parameter. The color may optionally start with a # character. This is helpful to mark a string as a color for easier readability. A complete color specification would be: `#AARRGGBB` where `AA` is two hexadecimal digits specifying an alpha value from `00` (transparent) to `FF` (opaque). `RR`, `GG` and `BB` specify red, green and blue values from `00` (none) to `FF` (full) brightness. A fully opaque color may be truncated to as few as 1 character. For example, `#ff` would indicate pure blue, and `#ff00` would be pure green. If the color requires an alpha value other than `FF` (opaque) then all 8 characters of the color must be specified. Thus, `#800000ff` would be 50% transparent blue. `#00000000` would be fully transparent black, which would be invisible.

2.2.4. Customization

This DataHub API contains the complete Java source code for the Java applets and TCP connection classes. You can create your own applets to display data in any form that you like, using this DataHub API source as a starting point.

2.2.5. Browser Limitations

Internet Explorer versions 6 and 7 cannot display a large number of Java applets on a single page. The limit is variable, but is in the vicinity of 15 applets per page. Mozilla Firefox can reliably display around 200 applets per page, but it may crash with larger numbers of applets on a page.

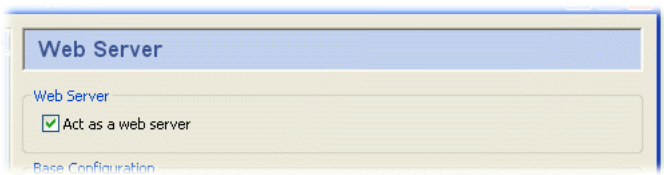
You can work around browser limitations by using the `DataHubTable` applet, or by designing your own applets to display more information per applet, thereby reducing the total number of applets on a page.

2.3. Configuring the DataHub Web Server

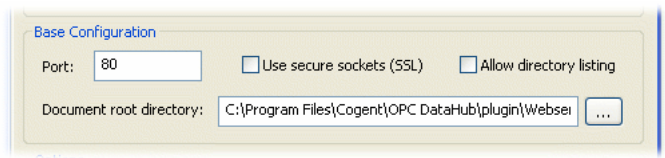
To configure the DataHub Web Server, follow these steps:

1. With the DataHub running, right click on the DataHub system-tray icon and choose Properties.
2. In the Properties window, select Web Server 

.
3. Check the Act as web server box.



4. The DataHub Web Server is preconfigured to run on port number 80, but you might need to change that setting in the Base Configuration section:



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.

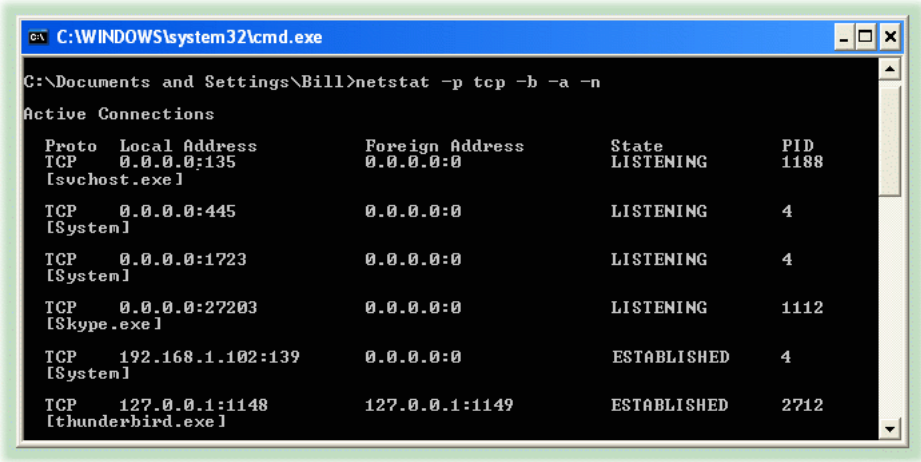
To get a list of which ports are in use on your machine, follow these steps:

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
```



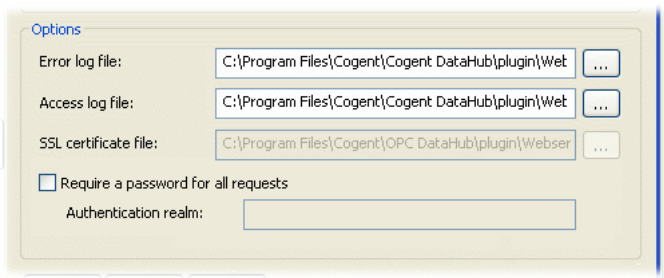
The result is a table showing the tcp protocol and executable name of all programs in use. There are two columns of interest: `Local Address` and `State`. In `Local Address`, the numbers at the end (after the colon) are the port number that the process is using. The `State` column shows the state of 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.

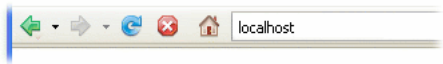
Authentication realm

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

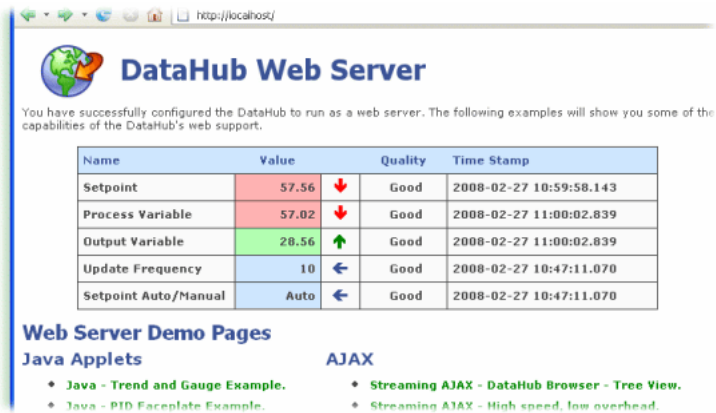
2.4. Testing the DataHub Web Server

To view a page served by the DataHub, open a web browser and type in the page URL. There is a demo page that comes with your DataHub installation where you can view data coming from DataSim. Here's how to access the page:

1. Ensure that the DataHub is running and is configured to act as a web server.
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 demos use data from DataSim, so DataSim must be running and connected for the data to update live.



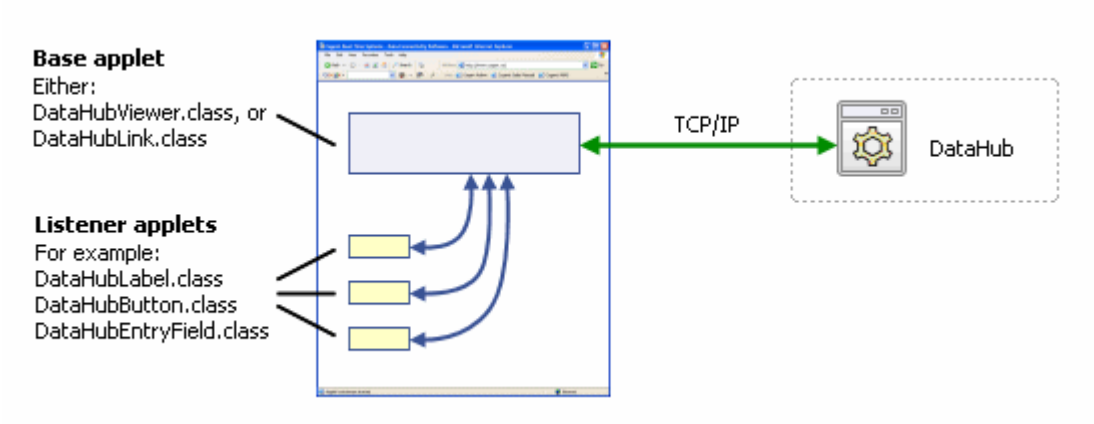
2.5. In the web page

You can display live data from your DataHub application in a standard web page by embedding Java applets that connect over the Internet to a DataHub running the Web Server. The DataHub distribution comes with a collection of Java applets, which can be viewed from the DataHub's Web Server demo page. These applets are documented in Chapter 3, *Standard Java Applets*. Embedding Java applets in

your HTML code is a simple matter of defining APPLET code as explained below, and editing the parameters to suit your needs.

2.5.1. Data model

Our model for displaying dynamic data in a web page uses a *Base* applet and *Listener* applets to reduce network bandwidth requirements and increase the speed of updates in the page.



The Base Applet

In our model, any web page that connects to the DataHub needs to include at least one applet derived from the `DataHubBaseApplet` class. We call this applet the Base applet. The Base applet makes the connection to the DataHub and receives the live data from the DataHub. In addition to displaying data, Base applets also relay dynamic data to other applets in the web page, which we call Listener applets (see below). You can have as many Base applets as you like in a page, but each one will require a separate connection to the DataHub. We recommend using the miniumum, which is one per DataHub domain. If your data comes from multiple domains, you might consider creating a new domain and bridging all of those points to that domain.

There are two Base applets to choose from in the Java API. The first is `DataHubViewer`, which displays a table of dynamic data in a web page. The second is `DataHubLink`, which simply puts a small "Powered by Cogent" logo on the screen. Using these as examples, you can build your own Base applets to display the data from the DataHub in any way you like.

The purpose of the Base applets is threefold:

1. To establish the link to the DataHub.
2. To display dynamic data.
3. To relay dynamic data to Listener applets.

Listener Applets

Listener applets also connect to the DataHub, but they do so through the Base applet connection. The data coming from the DataHub goes to the Base applet first. From there it is transferred to all Listener applets on the same web page. This dramatically improves the speed of the connection and reduces network traffic. You can add as many Listener applets to the page as you like with little impact on data transfer rates. The recommended data model is to have one Base applet per page, feeding multiple Listener applets.

We have included a number of example Listener applets in the API that you can use to build dynamic web pages. Or you can extend these applets to make your own custom applets.

2.5.2. Example HTML coding

Embedding Java applets in a web page can be complicated by inconsistencies between different browsers. To simplify this process we have created a shortcut that reduces the amount of HTML code you have to write.

The `DataHubDummy` applet

Internet Explorer requires a Java plug-in to recognize and correctly display the `APPLET` tag. To have IE download and install the Java plug-in you must ensure it has the correct information about what version of the plug-in to install, and where to go to download it. This information is provided in the following `OBJECT` tag.

```
<OBJECT CLASSID="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
CODEBASE="http://java.sun.com/update/1.5.0/jinstall-1_5_0-windows-i586.cab#Version=1,5,0"
WIDTH="0" HEIGHT="0">
 <PARAM NAME="java_code" VALUE="cogent/DataHubDummy.class">
 <PARAM NAME="java_codebase" VALUE="/java/">
 <PARAM NAME="java_archive" VALUE="DataHub.jar">
</OBJECT>
```

If the Java plug-in has *already* been installed in the browser, then IE will simply load the `DataHubDummy` code and execute it. The code does nothing and simply returns control to the browser.

If the Java plug-in has *not* been installed, then this `OBJECT` code instructs IE where to go to download the plug-in. After the plug-in is installed, the Java code is run and control is once again returned to the browser.

After IE has installed the Java plug-in it will then be able to recognize the `APPLET` tag in future web pages it visits. We take advantage of this fact to eliminate the need to wrap each `APPLET` tag in a duplicate `OBJECT` tag, thus reducing the amount of HTML code needed to embed each applet.

Firefox, Mozilla and NetScape all recognize the `APPLET` tag and will load the Java plug-in (if needed) when they first encounter the `APPLET` tag in your web page.

We recommend placing this shortcut `OBJECT` tag in your web pages before any `APPLET` code definitions. This will ensure that IE recognizes the `APPLET` tags in the rest of your web page.

2.5.3. Using a Base applet

To make a connection to the DataHub, you need to use a Base applet, such as `DataHubViewer.class` or `DataHubLink.class`, somewhere in your page. Here is an annotated version of the code using `DataHubLink.class`:

```
<applet code="" width="5" height="5">                          ❶
   <param name="java_code" value="cogent/DataHubLink.class">   ❷
   <param name="java_codebase" value="/java">                  ❸
   <param name="java_archive" value="DataHub.jar">             ❹
   <param name="port" value="4600">                            ❺
   <param name="domain" value="DataPid">                       ❻
   <param name="name" value="BaseAppletT1">                    ❼
   Your browser is not java enabled.                           ❽
</applet>
```

❶ Instructs the browser to reserve space in the page for the applet.

❷ Calls the code of the widget class.

❸ Tells the web server where the `DataHub.jar` file for the code is located.

❹ Identifies the `DataHub.jar` file that contains the code. This same `DataHub.jar` file is used for all the classes in this API.

❺ Identifies the port number of the connection.

❻ Identifies the DataHub domain where the data resides.

❼ Assigns a name to this base applet. This name is used by any `DataHubListener` applets embedded in the same web page.

❽ Message displayed if the browser does not have Java enabled. The browser will attempt to download the appropriate Java plug-in. We use the `OBJECT` tag shortcut described above to instruct the browser where to find the Java plug-in.

2.5.4. Using Listener applets

Once a connection has been established with a `DataHubBaseApplet`, you connect `DataHubListener` applets to it to display and interact with your data. Here is an example of a Listener applet, `DataHubEntryField`.

```
<applet code="" width="40" height="20" align="absmiddle" hspace="10">
  <param name="java_code" value="cogent/DataHubEntryField.class" />
  <param name="java_codebase" value="/java" />
  <param name="java_archive" value="DataHub.jar" />
  <param name="bgcolor" value="faecaf" />
  <param name="fgcolor" value="000000" />
  <param name="name" value="DataHubItem" />
  <param name="points" value="PID1.Sp" />
  <param name="parent" value="BaseAppletT1" />
  <param name="editable" value="true" />
  <param name="min" value="0" />
  <param name="max" value="100" />
  <param name="numeric" value="true" />
  <param name="font" value="tahoma;;12" />
  <param name="halign" value="center" />
  <param name="valign" value="center" />
  <param name="format" value="0.0" />
  <param name="border" value="line;;1" />
</applet>
```

To customize this code, you just need to change the parameters, which are explained in the DataHubEntryField reference page.

Chapter 3. Standard Java Applets

DataHubBaseApplet

`DataHubBaseApplet` — a base class, not instantiated.

Description

This is a base class that should not be directly instantiated. It provides the facilities and parameters for making a connection to the DataHub to its derived classes. Every web page must contain one applet derived from `DataHubBaseApplet`. Currently the choices are `DataHubViewer` or `DataHubLink`.

Base Class

`Applet`

Parameters

Parameter	Type	Default	Description
name	`string`	`DataHubViewer`	A `DataHubBaseApplet` publishes a name that is used by `DataHubListener` applets to connect with the `DataHubBaseApplet`. This name can be any unique string.
domain	`string`	`DataSim`	This parameter determines the data domain in the DataHub that this web page will use. Currently all DataHub points on a web page must come from a single data domain. This applet will subscribe to every data point in the domain.
host	`string`	`null`	The host name running the DataHub. Normally you should not use this parameter. The `DataHubBaseApplet` will compute the host name from the URL of the page. If you provide this parameter you must also ensure that the Java permissions of your web browser are appropriate to allow the browser to connect to this host.
port	`integer`	`4600`	The port number that the DataHub is listening on as a master for mirroring (or tunnelling).
bgcolor	`color`	`white`	The background color of the applet.
fgcolor	`color`	`black`	The foreground color of the applet.
username	`string`	`null`	The user name used to authenticate this connection with the DataHub.
password	`string`	`null`	The password used to authenticate this connection with the DataHub.

DataHubBaseButton

`DataHubBaseButton` — a base class for buttons.

Description

This is the base class for `DataHubButton`, `DataHubCheckBox`, `DataHubButton`, and `DataHubRadioButton` applets. It should not be directly instantiated.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
bgcolor	color	white	The background color for this applet
fgcolor	color	black	The foreground color for this applet
text	string	Press	The text to display on the button.
bgcolor1	color	white	The background color used when the data point value is equal to *value1*.
fgcolor1	color	black	The foreground color used when the data point value is equal to *value1*.
value1	string	0	The data value that will cause the button to display in colors *bgcolor1*/*fgcolor1*.
text1	string	null	The text to display on the button when the data point value is *value1*. If this is not specified, it is given the value of the *text* parameter.
bgcolor2	color	white	The background color used when the data point value is equal to *value2*
fgcolor2	color	black	The foreground color used when the data point value is equal to *value2*
value2	string	1	The data value that will cause the button to display in colors *bgcolor2*/*fgcolor2*.
text2	string	null	The text to display on the button when the data point value is *value2*. If this is not specified, it is given the value of the *text1* parameter.
numeric	boolean	false	If this parameter is true, the data value will be compared to *value1* and *value2* using a numeric comparison. If false, the data value is assumed to be a string.
Inherited	-	-	All parameters from `DataHubListener`.

DataHubButton

`DataHubButton` — a push button that sends a value for a DataHub point.

Description

This applet displays a push button. When the button is pressed, it will toggle the data point value between *`value1`* and *`value2`* (inherited from `DataHubBaseButton`). If the data point value is neither *`value1`* nor *`value2`*, it will be set to *`value1`* when the button is pressed.

Base Class

`DataHubBaseButton`

Parameters

Parameter	Type	Default	Description
Inherited	-	-	All parameters from `DataHubBaseButton`.
Inherited	-	-	All parameters from `DataHubListener`.

DataHubCheckBox

`DataHubCheckBox` — a check box that toggles a DataHub point between two values.

Description

This applet displays a check box. When the check box is selected, it will set the data point value to *`value2`* (inherited from `DataHubBaseButton`). If the check box is unselected it will set the data point value to *`value1`*.

Base Class

`DataHubBaseButton`

Parameters

Parameter	Type	Default	Description
Inherited	-	-	All parameters from `DataHubBaseButton`.
Inherited	-	-	All parameters from `DataHubListener`.

DataHubEntryField

`DataHubEntryField` — an entry field for changing the value of a DataHub point.

Description

This applet displays an entry field. If the field is editable, this applet will write data into the DataHub.

Base Class

`DataHubListener`

Parameters

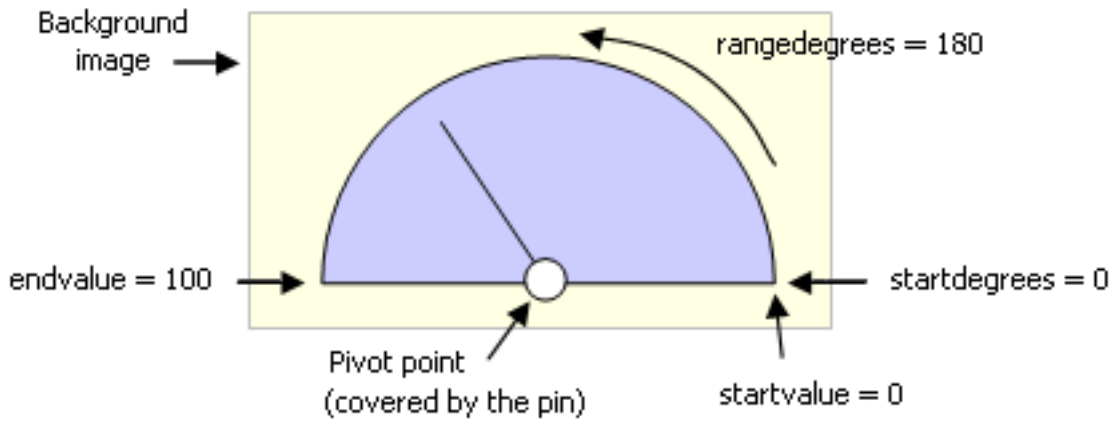
Parameter	Type	Default	Description
editable	boolean	false	If this is true, the field is editable. When the user types into this field, the value of the data point is modified in the DataHub.
numeric	boolean	false	If this is true, user entry is treated as a number, otherwise it is treated as string.
min	number	0	If this field is numeric, this specifies the minimum value allowed for user entry.
max	number	100	If this field is numeric, this specifies the maximum value allowed for user entry.
Inherited	-	-	All parameters from `DataHubListener`.

DataHubGauge

`DataHubGauge` — a gauge for displaying DataHub point values.

Description

This applet consists of a needle that moves in an arc against a fixed background to indicate values of a DataHub point.



Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
needlecolor	`Color`	`RED`	The color of the needle.
pincolor	`Color`	`RED`	The color of the pin, which is the dot at the pivot point of the needle.
needlelength	`int`	`50`	The length of the needle, in pixels.
basex	`int`	`0`	The x coordinates of the pivot point of the needle.
basey	`int`	`0`	The y coordinate of the pivot point of the needle.
pinradius	`int`	`5`	The size of the pin.
startvalue	`double`	`0`	The point value associated with the start of the arc through which the needle travels.
endvalue	`double`	`100`	The point value associated with the end of the arc through which the needle travels.
rangedegrees	`double`	`180`	The number of degrees of the arc through which the needle will travel.
startdegrees	`double`	`0`	The beginning point of the arc through which the needle will travel. Degrees start at a 3 o'clock position and increase in an anticlockwise direction. So a start point of zero and a range of `180` will give a half circle arc on the horizontal.

Parameter	Type	Default	Description
gaugeimage	string	null	The filename for a background image for the gauge.
editable	boolean	false	Allows the viewer to move the needle and change the value.
label1 to *label10*	string	null	You can add up to 10 text labels which will be displayed in front of the background image.
labelx1 to labelx10	int	0	The x coordinates for the center of each text label.
labely1 to labely10	int	0	The y coordinates for the center of each text label.
labelfont1 to labelfont10	Font	null	The font for each text label.
labelcolor1 to label-color10	Color	RED	The color for each text label.
valuex	int	0	The x coordinates of a label that shows the value associated with the data point.
valuey	int	0	The y coordinates of a label that shows the value associated with the data point.
valuefont	Font	null	The font for the value label.
valueformat	string	null	The format of the value label.
valuecolor	Color	RED	The color of the value label.
showvalue	boolean	false	Determines whether the value is shown or hidden.
step	double	0	The minimum value change necessary to reposition the needle.

DataHubLabel

`DataHubLabel` — displays the value of a DataHub point in a text field.

Description

This applet displays a data point value as a text field.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
`editable`	`boolean`	`false`	If this is true, the field is editable. When the user types into this field, the value of the data point is modified in the DataHub.
`format`	`string`	`null`	If the value of the data point is a number, this parameter specifies the format to be used. The format must be in the form expected by the Java class `DecimalFormat`.
Inherited	-	-	All parameters from `DataHubListener`.

DataHubLineChart

`DataHubLineChart` — displays a real-time trend of DataHub data.

Description

This is an experimental applet that displays a multi-trace real-time trend.

DataHubLink

`DataHubLink` — makes a TCP connetion to the DataHub.

Description

This applet is a `DataHubBaseApplet`, and therefore makes a TCP connection to the DataHub. It displays a box containing the words "`Powered by Cogent`" on a colored background. If the TCP link to the DataHub is connected, the background is green. If the link is disconnected, the background is red. The background is colored yellow while a connection attempt is in progress.

Base Class

`DataHubBaseApplet`

Parameters

Parameter	Type	Default	Description
Inherited	-	-	All parameters from `DataHubBaseApplet`.

DataHubListener

`DataHubListener` — a base class for all applets that send and receive data.

Description

`DataHubListener` is a base class for all applets that send and receive data with a `DataHubBaseApplet`. You should never instantiate a `DataHubListener` directly. This is the class that you would likely derive from when creating a custom applet for use with the DataHub.

Base Class

`Applet`

Parameters

Parameter	Type	Default	Description
border	string	null	One of two options: 1) "style;depth;highlight;shadow", where style is one of: bevel or etch, depth is one of: lowered or raised and highlight and shadow are colors; **or** 2) "line;color;thickness".
margins	string	"0;0;0;0"	The number of pixels determining the empty space between the outside border and the content of the applet. The order is: top;left;bottom;right.
font	Font	null	The font for the displayed characters. Font is specified as: font-name;bold|italic|normal;pointsize where bold|italic|normal means any of these, separated by the | character
valign	string	null	The vertical alignment of the applet in the space provided. One of "center", "top", or "bottom".
halign	string	null	The horizontal alignment of the applet in the space provided. One of "center", "left", or "right".
enabled	boolean	true	Enables or disables the Java component within the applet. For example, if you disable a scroll bar, it will no longer respond to the mouse.
parent	string	null	The same as the *name* parameter of the applet derived from `DataHubBaseApplet` on this web page.
points	string	null	A list of unqualified (i.e., without domain: prepended to them) data point names, separated by spaces. This parameter is used when the applet will service more than one data point.
point	string	null	If the *points* parameter is null, then this parameter will contain a single point name specifying the data point that this applet will service.
fgcolor	color	BLACK	The foreground color of this applet.

Parameter	Type	Default	Description
bgcolor	color	WHITE	The background color of this applet.
opaque	boolean	false	An optimization flag. If you specify that the applet is opaque, the redrawing functions in the Java engine can safely clip any windows behind this applet. If the applet contains a transparent or translucent background, then it is not opaque. In that case, setting the applet to opaque will cause drawing errors.
bgmode	string [scale, center, tile]	scale	A string specifying the tiling mode for the background image (`scale`, `center`, or `tile`).
format	string	null	For applets that display point values as text, this specifies the format to be used. The format must be in the form expected by the Java class `DecimalFormat`.
bgimage	string	null	A URL specifying a background image to display behind the table cells.
bgalpha	string	1.0	A string specifying the alpha blending value (`0.0` - `1.0`). Where `0` is transparent (invisible) and `1.0` is opaque. Numbers between `0` and `1` indicate degrees of translucency.

DataHubProgressBar

`DataHubProgressBar` — a horizontal or vertical progress bar for displaying DataHub data.

Description

This applet displays a horizontal or vertical progress bar. The data point for this applet should always have a numeric value.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
text	string	null	A text string associated with the bar.
orientation	string (horizontal or vertical)	horizontal	Specifies the orientation of the progress bar.
min	number	0	The lower bound for the range of the progress bar.
max	number	100	The upper bound for the range of the progress bar.
Inherited	-	-	All parameters from `DataHubListener`

DataHubRadioButton

`DataHubRadioButton` — a radio buttona special button, used in a DataHubRadioGroup.

Description

This applet displays a radio button. When the radio button is selected, it will set the data point value to *`value2`* (inherited from `DataHubBaseButton`). If the radio button is unselected it will set the data point value to *`value1`*. If you want to display multiple mutually exclusive radio buttons, use `DataHubRadioGroup` instead.

Base Class

`DataHubBaseButton`

Parameters

Parameter	Type	Default	Description
Inherited	-	-	All parameters from `DataHubBaseButton`
Inherited	-	-	All parameters from `DataHubListener`

DataHubRadioGroup

`DataHubRadioGroup` — a group of radio buttons for assigning a value to a DataHub point.

Description

This applet displays a group of radio buttons, allowing the user to select one value from a group of mutually exclusive values for a data point. You must specify the number of buttons in the group, and then specify the colors, text, value and numeric flag for each button on the group. A radio group acts on a single data point. Each radio button in a `DataHubRadioGroup` is an instance of a `DataHubRadioButton`. Radio buttons are arranged intelligently based on the width and height of the applet.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
buttoncount	integer	0	Specifies the number of radio buttons in this button group.
bgcolor_1	color	white	The background color of the first radio button.
fgcolor_1	color	black	The foreground color of the first radio button.
text_1	string	Press	The text to display on the first radio button.
value1_1	string	0	The data value associated with the first radio button.
numeric_1	boolean	false	A flag indicating that the first radio button should treat its data value as numeric.
bgcolor_N	color	white	The background color of the *N*th radio button.
fgcolor_N	color	black	The foreground color of the *N*th radio button.
text_N	string	Press	The text to display on the *N*th radio button.
value1_N	string	0	The data value associated with the *N*th radio button.
numeric_N	boolean	false	A flag indicating that the *N*th radio button should treat its data value as numeric.
Inherited	-	-	All parameters from `DataHubListener`

DataHubSlider

`DataHubSlider` — a slider that assigns a numeric value to a DataHub point.

Description

This applet displays a slider that allows the user to select a numeric value by interactively dragging a slider handle. The resulting value will be written to the DataHub. The data point for this applet should always have a numeric value.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
showticks	boolean	false	Shows the tick marks on the slider.
showtrack	boolean	true	Shows a track for the pointer.
inverted	boolean	false	Inverts the maximum and minimum values.
snaptoticks	boolean	false	Causes the handle to snap to the tick marks while dragging.
majortick	double	50	The numerical increment for each major tick.
minortick	double	10	The numerical increment for each minor tick.
orientation	string (horizontal or vertical)	horizontal	The orientation of the slider, one of `horizontal` or `vertical`.
min	number	0	The lower bound for the range of the slider.
max	number	100	The upper bound for the range of the slider.
Inherited	-	-	All parameters from `DataHubListener`

DataHubSpinner

`DataHubSpinner` — sets the value of a DataHub point using up and down arrows.

Description

This applet displays a numeric entry field with up and down arrows beside it. The user is able to select a value by either entering a number into the field, or by pressing the up and down arrows to specify a value. The resulting value will be written to the DataHub. The data point for this applet should always have a numeric value.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
step	number	1	Specifies the amount by which the value will change when the user presses the up or down arrows.
min	number	0	The lower bound for the value.
max	number	100	The upper bound for the value.
type	string (number, list or date)	number	The type of data to select. Currently only the number type is functional.
Inherited	-	-	All parameters from `DataHubListener`

DataHubTable

`DataHubTable` — displays multiple DataHub data points in a single applet.

Description

This is an experimental applet that displays data in a tabular format, similar to a spreadsheet display. The purpose of this applet is to display large amounts of information in rows and columns while still using only a single applet in a web page. This works around limitations in the browser if the browser is unable to display many applets at once.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
format	string	null	The default number format for cells in the table.
selectable	boolean	false	Determines whether cells in the table can be selected with the mouse.
rows	number	10	The number of table rows.
cols	number	2	The number of table columns.
grid	boolean	false	Determines whether the table will display grid lines around the cells.
gridcolor	color	null	Specifies the grid color.
header	boolean	true	Determines whether the table will have a header containing resizable column headers.
lastcell	number	100	The maximum cell number to search when configuring cell contents (see Cell Parameters).
lastclass	number	100	The maximum cell class number to search when configuring cell classes (see Cell Parameters).
lastrow	number	100	The maximum row height number to search when specifying row heights (see Cell Parameters).
widths	string	null	A delimited string containing the pixel widths of all of the columns in the table (see Delimited Strings).
colnames	string	null	A delimited string containing the column names for all of the columns in the table (see Delimited Strings).
doublebuffer	boolean	false	Determines whether the table should be displayed using double-buffer rendering. This eliminates flicker at the cost of higher CPU usage.
cellN	string	null	The *N*th cell specification (see Cell Parameters).
fontN	string	null	The *N*th cell font specification (see Cell Parameters).
vmapN	string	null	The *N*th cell value map (see Cell Parameters).
classN	string	null	The *N*th class specification (see Cell Parameters).
cfontN	string	null	The *N*th class font specification (see Cell Parameters).

Parameter	Type	Default	Description
cvmapN	string	null	The *N*th class value map (see Cell Parameters).
rowheightN	string	null	The *N*th row height specification (see Cell Parameters).
Inherited	-	-	All parameters from DataHubListener

Cell Parameters

The content of the DataHubTable is specified on a cell-by-cell basis. Each non-empty cell has one or more parameters describing its content. These parameters are:

- Cell specification
- Cell font information
- Cell value map

The cell specification is mandatory for each non-empty cell. The font information and value map are both optional.

A cell can be a member of a cell class. This allows you to specify many of the attributes of a cell once, and then repeat that many times for cells in the table. You can then make changes to the class to modify the rendering of all of the cells in the class at once.When the DataHubTable is initially rendered, it searches its parameter list for cell specifications. These are numbered parameters, starting at 1. For example, parameter *cell1* is the first non-empty cell, and *cell37* could be the 37th non-empty cell. The parameter numbers (1 and 37 in this case) must be unique, but not necessarily contiguous. The DataHubTable will search its parameter space from 1 to *lastcell* (by default 100) looking for cell specifications. You can change the value of *lastcell* to broaden the range of acceptable cell numbers.

The content of a cell can be either a string or an image. A string can be the value of a DataHub point, a constant string, or a range-mapped value. A range map uses a DataHub point value to compute the string or image that the cell will display. This is done by specifying a series of point value ranges and the cell classes that will be used to display the cell for each range. As the point value changes, the cell class used to render that cell will be computed based on the range map. For example, you may wish to display the word false when a point value is zero and true when the point value is one.

Delimited Strings

The specification strings for the *cellN*, *fontN*, *vmapN*, *classN*, *cfontN* and *cvmapN* all consist of a string of arguments, separated by delimiters. By default the delimiter character is a semicolon (;). If you wish to have a semicolon as part of any argument in the specification string, you must choose another delimiter. If the specification string starts with any non-alphanumeric character, that character is taken to be the delimiter for the arguments in the string. For example the string one;two:two;3;4 consists of four arguments, separated by semicolons, the default delimiter. The second argument is the string two:two. To specifiy an argument that contains a semicolon, you can choose another delimiter, such as ~one~two;two~3~4. Here the delimiter is the ~ character. Remember that DataHub point names contain colon (:) and decimal (.) characters, so these are not typically good choices for delimiters.

Setting Row Heights

It is possible to set the height of an individual row in the table. This is done by specifying one or more *rowheightN* parameters, where *N* is an ordinal number starting at 1. The value of this parameter is a delimited string containing the row number, starting at 1, and the target height in pixels. By default the

table searches for row height specifications in the range of `1` to `100`. If you would like to specify ordinal numbers for *`rowheightN`* above `100`, set the table parameter *`lastrow`* to a larger number.

Evaluation Order

The `DataHubTable` does not evaluate its parameters in the order that they appear in the applet definition in your HTML file. The parameter list is searched in a specific order to ensure that class definitions precede cell definitions, and that font and value map parameters are matched to their appropriate cell or class definitions. Parameters that have ordinal numbering (*`cell`*, *`class`*, *`vmap`*, *`cvmap`*, *`font`*, *`cfont`*, *`rowheight`*) will always be evaluated from lowest to highest ordinal number, regardless of their position in the parameter list of the applet.

The consequence of this evaluation order is that it does not matter how you arrange the parameters within your HTML file. The order of specification will not affect the order of evaluation.

DataHubTable Cell Classes

`DataHubTable Cell Classes` — assign identical arguments to many table cells.

Description

A cell class is a means to assign the same rendering arguments to many different cells. The cell class specifies all of the arguments for a cell except for row, column, label and point name. Consequently, to add a cell that has the same arguments as an existing class, you must only specify those arguments along with the name of the class (see Member of Class - Type 3). A cell class is very convenient if you plan to have many cells using the same rendering. If you decide to change the rendering arguments for those cells, they can all be changed at once by changing the definition of the class. A cell class is also required to provide the rendering arguments for a cell when that cell has a value map. The value map refers to a different cell class for each of the ranges in the value map.

Cell classes are specified similarly to cells. They consist of up to three definitions for rendering, font and value map. If a class contains a value map, then the point value will be used to select a class through the value map. If that class also has a value map, the point value will be used again to select a class from the value map of the referred class. This class reference chain will resolve to an indefinite depth. Be sure not to create classes that refer to one another directly or indirectly through chained value maps. Cell class font specifications are identical to cell font specifications, except that the parameter name is `cfont` instead of `font`.

Cell class value map specifications are identical to cell value map specifications, except that the parameter name is `cvmap` instead of `vmap`.

A cell class has a type, much like a cell. The cell class can be of type 0 (label), 1 (point) or 4 (image). Label and image types are static. That is, the resulting contents of the cell do not change. In order to modify the contents of a label or image type, you must modify the class of the cell through a value map.

Label (Static Text) - Cell Class Type 0

A label specifies a text string that does not change.

Argument	Type	Default	Description
`name`	`string`	`required`	The name of this cell class.
`type`	`number`	`required`	0 for a label cell.
`xalignment`	`string`	`left`	`left`, `center` or `right` horizontal alignment of the text in the cell.
`yalignment`	`string`	`center`	`top`, `center`, or `bottom` vertical alignment of the text in the cell.
`fgcolor`	`color`	`table default`	The color used to render the text. If not specified, uses the table's *`fgcolor`* parameter.
`bgcolor`	`color`	`table default`	The color used to render the cell background. If not specified, uses the table's *`bgcolor`* parameter.

Example

```
<PARAM NAME="class1" VALUE="myclass;0;left;center;#ffff00;#0000ff">
<PARAM NAME="cell1" VALUE="2;3;3;Hello World;myclass">
```

Will render the words `Hello World` with left/center justification in yellow text on a blue background.

DataHub Point Value - Cell Class Type 1

This cell will display the value of an DataHub point. If the point value is a number you can specify the decimal format with which it is displayed.

Argument	Type	Default	Description
name	string	required	The name of this cell class.
type	number	required	1 for DataHub point value.
xalignment	string	left	left, center or right horizontal alignment of the text in the cell.
yalignment	string	center	top, center or bottom vertical alignment of the text in the cell.
fgcolor	color	table default	The color used to render the text. If not specified, uses the table's *fgcolor* parameter.
bgcolor	color	table default	The color used to render the cell background. If not specified, uses the table's *bgcolor* parameter.
format	string	table default	The decimal format to use when the value of the point is a number

Example

```
<PARAM NAME="class1" VALUE="myclass;1;left;center;#ffff00;#0000ff">
<PARAM NAME="cell1" VALUE="2;3;3;Sine;myclass">
```

Will render the value of the point Sine in cell (2, 3) with left/center justification in yellow text on a blue background, with 1 to 3 decimal places of accuracy.

Image - Cell Class Type 4

An image cell renders an image into the cell, clipping the image at the borders of the cell. The image is obtained from a URL.

Argument	Type	Default	Description
name	string	required	The name of this cell class.
type	number	required	4 for Image.
xalignment	string	left	left, center or right horizontal alignment of the text in the cell.
yalignment	string	center	top, center or bottom vertical alignment of the text in the cell.
fgcolor	color	table default	The color used to render the cell foreground. If not specified, uses the table's *fgcolor* parameter. Foreground color is unused for image type cells.
bgcolor	color	table default	The color used to render the cell background. If not specified, uses the table's *bgcolor* parameter.
imageurl	string	required	The URL from which to obtain the image. This may be either an absolute URL of the form http://domain/path/image, or a relative URL of the form /path/image or path/image.
alpha	number	1.0	An alpha blend value from 0.0 to 1.0 where 0.0 is transparent and 1.0 is opaque.

Example

```
<PARAM NAME="class1" VALUE="checkmark;4;center;center;#0;#00000000;checkmark.gif;0.5">
<PARAM NAME="cell1" VALUE="2;3;3;null;checkmark">
```

Will render the image from the URL `checkmark.gif` centered in the cell at `(2, 3)`. The image will be semi-transparent. The background of the cell will be transparent.

DataHubTable Cell Specification

`DataHubTable Cell Specification` — specifies the position and contents of a table cell.

Description

A cell specification is a delimited string specifying the position and contents of a non-empty cell. Part of the specification is a content type. The arguments in the cell specification depend on the content type. The arguments must be specified in the order listed in the tables below. If you wish to specify an optional argument, you must also specify all arguments preceding it. If you wish to use the default value for an optional argument, specify `null` for that argument.

Label (Static Text) - Cell Type 0

A label specifies a text string that does not change.

Argument	Type	Default	Description
row	color	required	The cell row, starting at 1.
column	color	required	The cell column, starting at 1
type	number	required	0 for a label cell.
label	string	required	The text to display in the cell.
xalignment	string	left	`left`, `center` or `right` horizontal alignment of the text in the cell.
yalignment	string	center	`top`, `center` or `bottom` vertical alignment of the text in the cell.
fgcolor	color	table default	The color used to render the text. If not specified, uses the table's *fgcolor* parameter.
bgcolor	color	table default	The color used to render the cell background. If not specified, uses the table's `bgcolor` parameter.

Example

```
<PARAM NAME="cell1" VALUE="2;3;0;Test;left;center;#ffff00;#0000ff">
```

Will render the word `Test` in cell `(2, 3)` with left/center justification in yellow text on a blue background.

DataHub Point Value - Cell Type 1

This cell will display the value of a DataHub point. If the point value is a number you can specify the decimal format with which it is displayed.

Argument	Type	Default	Description
row	color	required	The cell row, starting at 1.
column	color	required	The cell column, starting at 1.
type	number	required	1 for DataHub point value.
pointname	string	required	The unqualified point name (no domain: part) to display.
xalignment	string	left	`left`, `center` or `right` horizontal alignment of the text in the cell.

Argument	Type	Default	Description
valignment	string	center	top, center or bottom vertical alignment of the text in the cell.
fgcolor	color	table default	The color used to render the text. If not specified, uses the table's *fgcolor* parameter.
bgcolor	color	table default	The color used to render the cell background. If not specified, uses the table's *bgcolor* parameter.
format	string	table default	The decimal format to use when the value of the point is a number.

Example

```
<PARAM NAME="cell1" VALUE="2;3;1;Sine;left;center;#ffff00;#0000ff;0.0##">
```

Will render the value of the point Sine in cell (2, 3) with left/center justification in yellow text on a blue background, with 1 to 3 decimal places of accuracy

Indirectly Formatted Point - Cell Type 2

An indirectly formatted point reads its cell specification from a DataHub point. The value of the point must be a string containing a valid value specification for a cell.

Argument	Type	Default	Description
row	color	required	The cell row, starting at 1.
column	color	required	The cell column, starting at 1.
type	number	required	2 for indirectly formatted point.
specpoint	string	required	The name of a DataHub point from which to read the cell specification for this cell.
valuepoint	string	null	If the cell specification in specpoint contains the word null for its point name, use valuepoint instead as the point from which to take the cell value.

Example

```
<PARAM NAME="cell1" VALUE="2;3;2;myspec;Sine">
```

will render the value of the point Sine in cell (2, 3) according to a cell's value specification contained in the DataHub point myspec. The point should contain just the content of a value parameter, like this:

```
2;3;1;Sine;left;center;#ffff00;#0000ff;0.0##
```

Member of Class - Cell Type 3

A class member takes its cell specification from a named cell class. All members of a cell class are formatted identically. Changes to the cell class will cause all cells in the class to be changed.

Argument	Type	Default	Description
row	color	required	The cell row, starting at 1.
column	color	required	The cell column, starting at 1.
type	number	required	3 for member of a class.

Argument	Type	Default	Description
label	string	required	If the cell class is type 0 (a label) then this is a static text string. If the cell class is type 1 (a point) then this is the name of the DataHub point.
classname	string	required	The name of the class to which this cell belongs.

Example

```
<PARAM NAME="cell1" VALUE="2;3;3;Sine;float_class">
```

Will render the value of the point `Sine` in cell `(2, 3)` according to the class specification of the class `float_class`.

Image - Cell Type 4

An image cell renders an image into the cell, clipping the image at the borders of the cell. The image is obtained from a URL.

Argument	Type	Default	Description
row	color	required	The cell row, starting at 1.
column	color	required	The cell column, starting at 1
type	number	required	4 for image.
imageurl	string	required	The URL from which to obtain the image. This may be either an absolute URL of the form `http://domain/path/image`, or a relative URL of the form `/path/image` or `path/image`.
alpha	number	`1.0`	An alpha blend value from `0.0` to `1.0` where `0.0` is transparent and `1.0` is opaque.

Example

```
<PARAM NAME="cell1" VALUE="2;3;4;checkmark.gif;0.5">
```

Will render the image from the URL `checkmark.gif` into the cell. The image will be semi-transparent.

DataHubTable Fonts

`DataHubTable Fonts` — specify the fonts for a table cell.

Description

The font for a cell is specified separately from the position and content. Each font specification is matched to the cell specification using the cell number. That is, if the cell `PARAM` name is `cell14` then the corresponding font specification will be `font14`. If a cell does not have a font specification then the default Java font is used.

A font specification consists of a delimited string with the following arguments:

Argument

Argument	Type	Default	Description
fontname	string	required	The name of the font, logical font, or font face. Uses the Java font names to resolve the font.
fontstyle	string	required	One of `normal`, `bold`, `italic` or `bolditalic`.
fontsize	number	required	The point size of the font.

Example

```
<PARAM NAME="font1" VALUE="Helvetica;bold;16">
```

DataHubTable Value Maps

`DataHubTable Value Maps` — change cell formatting or text based on a DataHub point value.

Description

A value map is a mechanism for changing the formatting or displayed text in a cell based on the value of a DataHub point. The value map consists of a series of ranges associated with cell classes. If the value of the point falls within a range, then the cell is rendered using the class associated with that range. Ranges are specified as a minimum and a maximum value. If the point value is greater than the minimum, or less than or equal to the maximum (i.e., minimum < value <= maximum), then the value is within the range. If the minimum and the maximum for a range are the same value, then the point value must exactly match the minimum to be within the range. Ranges are compared to the value in the order in which they appear in the value map. Overlapping ranges are allowed. The first range to match the point value will be used.

A value map is a delimited string consisting of one or more groups of three arguments. The arguments in each group are *`minimum;maximum;classname`* respectively. The groups are delimited by semicolons (`;`), and there can be any number of such groups. If the value for any minimum is `inf` then the minimum is taken to be negative infinity. If the value for any maximum is `inf` then the maximum is taken to be positive infinity.

If no range matches the input value then the value is rendered in the default font, format and colors.

To assign a value map to an individual point, use a parameter named `vmap`*`N`* where *`N`* is the ordinal number associated with the cell specification.

To assign a value map to a class, use a parameter named `cvmap`*`N`* where *`N`* is the ordinal number associated with the cell class specification.

Example

```
<PARAM NAME="vmap1" VALUE="inf;1;false_class;1;inf;true_class">
```

Will render a boolean value in either `false_class` or `true_class` depending on whether its value is `0` or `1`.

```
<PARAM NAME="vmap2" VALUE="inf;-10;red_class;-10;-5;yellow_class;-5;5;
              green_class;5;10;yellow_class;10;inf;red_class">
```

Will render a numeric value as `red_class` if its absolute value is greater than `10`, as `yellow_class` if its absolute value is between `5` and `10`, and as `green_class` if its absolute value is less than `5`.

DataHubToggleButton

`DataHubToggleButton` — toggles a DataHub point between two values.

Description

This applet displays a toggle button. When the toggle button is selected, it will set the data point value to *`value2`* (inherited from `DataHubBaseButton`). If the toggle button is unselected it will set the data point value to *`value1`*.

Base Class

`DataHubBaseButton`

Parameters

Parameter	Type	Default	Description
Inherited	-	-	All parameters from `DataHubBaseButton`
Inherited	-	-	All parameters from `DataHubListener`

DataHubTrend

`DataHubTrend` — a dynamically updating graphical display of DataHub values.

Description

This applet displays the values of several DataHub points in a dynamically updating graphical display.

Base Class

`DataHubListener`

Parameters

Parameter	Type	Default	Description
`maxpoints`	`int`	`0`	The maximum number of points per trace to display. This determines the amount of data in the trace, which will have an effect on CPU usage. Do not set both *`maxpoints`* and *`maxtime`*.
`maxtime`	`double`	`20`	The number of seconds worth of data per trace to display. This determines the amount of data in the trace, which will have an effect on CPU usage. Do not set both *`maxtime`* and *`maxpoints`*.
`refreshrate`	`int`	`100`	The refresh rate of the display, in milliseconds.
`min`	`double`	`0`	The Y-axis minimum.
`max`	`double`	`100`	The Y-axis maximum. (Note that the X-axis span is automatically determined by the minimum and maximum time stamps for all of the traces in the trend.)
`pointN`	N/A	none	Specifies which DataHub point to plot, as well as the color and shape of the trace. You can plot a maximum of 10 traces. See below for details.

The *`pointN`* parameter

You will need to specify a *`pointN`* parameter for each trace in the trend, as follows:

Syntax

```
<param name="pointN" value="pointname;color;flags">
```

Name and Value

`point`*`N`*

The first trace must be named `point1`, the second trace `point2`, the third `point3` and so on up to `point10`.

`pointname`

The name of a DataHub point, without the domain name.

`color`

A color, specified as `#000000`.

`flags`

(optional) One of: `extend`, `square`, or `extend|square` where:

`extend`

The trace for this point will be automatically extended horizontally on each update of the other traces in the trend graph.

`square`

The trace will be drawn as a step function instead of drawing a straight line from one value to the next.

Example

```
<param name="point1" value="PID1.Mv;#ff9900">
<param name="point2" value="PID1.Sp;#0000ff;extend|square">
<param name="point3" value="PID1.Pv;#009900">
```

DataHubViewer

`DataHubViewer` — makes a TCP connection to the DataHub, displaying all available data in a table.

Description

This applet is a `DataHubBaseApplet`, and therefore makes a TCP connection to the DataHub. It displays a four-column table in the browser showing the current values of every data point it is managing. The status of the connection is shown in text above the first line of the table.

Base Class

`DataHubBaseApplet`

Parameters

Parameter	Type	Default	Description
colwidths	string	null	If specified, this is a list of 4 numbers separated by a single space character. Each number specifies the width in pixels of the corresponding column in the data table.
login	boolean	false	If true, the applet will display a user name and password dialog, and will send authentication information to the DataHub.
Inherited	-	-	All parameters from `DataHubBaseApplet`.

Chapter 4. The `DataHubConnector` Class

4.1. Overview

Syntax

For C++ (Windows MFC, QNX, Linux):

```
class CDataHubConnector : public CWnd
```

For C++ (Windows ATL):

```
class CDataHubConnector : public
           CWindowImpl<CDataHubConnector,CWindow,CFrameWinTraits>
```

For Java:

```
public class DataHubConnector
```

For C#:

```
public class DataHubConnector
```

Remarks

This class provides the base functionality for a client to connect to the DataHub. The constructor for this class is `DataHubConnector`. The destructor for this class is `~DataHubConnector`. The methods for this class are arranged by category in the Categorized List of Methods, and alphabetically in the `DataHubConnector` Methods reference.

Requirement Statements

For C++:

```
#include <CDataHubConnector.h>
```

For Java:

```
import cogent.DataHubConnector;
```

For C#:

```
using Cogent.DataHubAPI;
```

4.2. Categorized List of Methods

These are most of the methods of the `DataHubConnector` class. The remaining methods, which are used for making callbacks, are presented in the following section.

Status Functions

`getCnxState` - retrieves the operational state of the connector object.
`getCnxStateString` - retrieves a string corresponding to the operational state.
`getCnxSubStateString` - provides detailed information on the connection state (C++ only).
`getErrString` - retrieves the last error string (C++ only).

Connection Control

`setReconnectionDelay` - sets the delay time between reconnection attempts.

`getReconnectionDelay` - retrieves the delay time between reconnection attempts.
`startReconnectionTimer` - starts the delay timer for reconnection attempts.
`cancelReconnectionTimer` - stops the delay timer for reconnection attempts.
`setHeartbeatTimes` - sets the period of the heartbeat and timeout timers.
`getHeartbeat` - retrieves the heartbeat timer period.
`getTimeout` - retrieves the timeout timer period.
`startHeartbeatTimers` - starts the heartbeat and timeout timers.
`cancelHeartbeatTimers` - cancels the heartbeat and timeout timers.
`activeHeartbeatTimers` - determines if both heartbeat timers are active.
`setConnectionParms` - sets the connection parameters.
`getHostName` - retrieves the host name connection parameter.
`getServiceName` - retrieves the port service name connection parameter (C++ only).
`isConnecting` - indicates whether a connection attempt is in progress.
`isConnected` - indicates whether a connection has been established.
`openConnection` - attempts to establish a connection to the DataHub.
`retryConnection` - opens a new connection to the DataHub (C++ only).
`closeConnection` - closes the connection to the DataHub.
`shutdown` - prepares for an application shutdown or disconnect (Java and C# only).

Messages

`sendLispMessage` - sends a message to the DataHub (C++ only).
`writeCommand` - sends a command to the DataHub (Java and C# only).
`escapedString` - prepares a string for use with `writeCommand` (Java and C# only).

Point Handling

`initializePointCache` - initializes local point cache usage.
`lookupPoint` - accesses a point from the point cache.
`registerDomain` - registers to receive updates from domain points.
`setDefaultDomain` - sets the default domain.
`getDefaultDomain` - returns the current default domain.
`registerPoint` - registers to receive updates from a DataHub point.
`unregisterPoint` - stops receiving updates from a DataHub point.
`createPoint` - creates a DataHub point.
`writePoint` - writes a new value to a DataHub point.
`readPoint` - gets the value of a DataHub point.
`setPointLock` - sets the lock attributes of a DataHub point.
`setPointSecurity` - sets the security attributes of a DataHub point.
`appendPointValue` - appends a string to a DataHub point.
`addPointValue` - adds a specified amount to a DataHub point value.
`multiplyPointValue` - multiplies a DataHub point value by a specified amount.
`dividePointValue` - divides the value of the named point by the specified value.

4.3. Making Callbacks

There are several callbacks associated with the `DataHubConnector` class. In C++ these are methods of the `DataHubConnector` class itself, while in Java and .NET they are methods of a separate interface, the `DataHubEventConsumer` interface.

Status Changes

`onStatusChange` - a virtual method invoked on change of status.

Connections

`onConnectionSuccess` - a virtual method invoked when a connection is established.
`onAlive` - a virtual method invoked on receipt of a heartbeat from the DataHub.

`onConnectionFailure` - a virtual method invoked when a connection or attempt to connect fails.

Message Receipts

`onAsyncMessage` - a virtual method invoked on receipt of a DataHub message.
`onSuccess` - a virtual method invoked on receipt of a success message.
`onError` - a virtual method invoked on receipt of an error message.

Point Changes

`onPointChange` - a virtual method invoked on receipt of a point value change.
`onPointEcho` - a virtual method invoked on receipt of a locally changed point value.

Chapter 5. The `DataHubPoint` Class

The `DataHubPoint` class represents a DataHub point object.

5.1. Overview

Syntax

For C++:

```
class CDataHubPoint
```

For Java:

```
public class DataHubPoint
```

For C#:

```
public class DataHubPoint
```

Remarks

DataHub point objects are the fundamental objects used to write, receive and manipulate data in the DataHub, via the `DataHubConnector` class. The `DataHubPoint` class provides a rich set of facilities to create, modify and inspect these objects.

DataHub points possess the following properties:

`value`

A value whose type is one of `PT_TYPE_STRING`, `PT_TYPE_REAL` (a double) or `PT_TYPE_INT32` (an int). The value is stored in a corresponding format, and can be converted by the various utilities to access the value.

`quality`

Indicates whether the DataHub has been updated with actual data or if a point is uninitialized. This is typically either `PT_QUALITY_GOOD` or `PT_QUALITY_BAD`. Connection status can also affect the point quality.

`confidence`

A user defined value, typically in the range of 0-100%. This can be used to model 'aging' of a point, and support 'fuzzy math' algorithms.

`timestamp`

Tags the real-time origin of the point as it is distributed. Typically this is set by the software module originating the value of a point. It is modelled as seconds and nanoseconds, providing a resolution that is limited only by the OS.

`userdata`

Allows the user to associate with a specific point whatever object may be useful to the application. This is primarily used by the point cache cability provided by the `DataHubConnector` class (see `initializePointCache`).

`locked`

Controls access to the point.

`security`

Controls access to the point.

`flags`

For internal use.

Requirement Statements

For C++:

```
#include <CDataHubPoint.h>
```

For Java:

```
import cogent.DataHubPoint;
```

For C#:

```
using Cogent.DataHubAPI;
```

See also

Categorized List of Methods, `DataHubConnector`

5.2. Categorized List of Methods

Constructors/Destructors

`DataHubPoint` - constructs a DataHubPoint object in various ways.
`~DataHubPoint` - destroys a DataHubPoint object.

General Methods

`clear` - clears the point.
`getName` - retrieves the point name.
`setName` - assigns a name to the point.
`qualifyName` - creates a point name string qualified by a domain name.
`unqualifyName` - removes the domain name qualifier from a point name.

Point Data Access Methods

`getType` - retrieves the point data type.
`setValue` - sets the point data to the specified type and value.
`setValueFromString` - sets the point data to the value represented by a string.
`getDoubleValue` - retrieves the point data as a double-typed value.
`getIntValue` - retrieves the point data as an int-typed value.
`getStringValue` - retrieves the point data as a string.

Point Information Access Methods

`setInfo` - sets the information properties of a point.
`setQuality` - sets the point quality.
`getQuality` - retrieves the point quality.
`getQualityString` - generates a string representing the point quality.
`setConfidence` - sets the user's confidence in a point.
`getConfidence` - retrieves the user's point confidence.
`setTimeStamp` - sets the point timestamp in various ways.
`getSeconds` - retrieves the timestamp seconds component.
`getNanoseconds` - retrieves the timestamp nanoseconds component.
`getDateString` - generates a 'standard' timestamp data/time representation.

`getListeners` - retrieves listeners on the point (Java only).
`removeListener` - removes a listener from the point (Java only).
`setLocked` - sets the locked property of the point.
`getLocked` - retrieves the locked property of the point.
`setSecurity` - sets the security property of the point.
`getSecurity` - retrieves the security property of the point.
`setFlags` - sets the flags property of the point.
`getFlags` - retrieves the flags property of the point.
`setUserdata` - associates the point with a user object.
`getUserdata` - retrieves the user object associated with the point.

Operators

`operator=` - assigns a new value to a DataHubPoint object (C++ only).

Appendix A. GNU General Public License

GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program",

below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE

PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type “show w”. This is free software, and you are welcome to redistribute it under certain conditions; type “show c” for details.

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than “show w” and “show c”; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program “Gnomovision” (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B. GNU Lesser General Public License

GNU Lesser General Public License

Version 2.1, February 1999

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. 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. DataHubConnector Methods

Table of Contents

These are the methods associated with the `DataHubConnector` class, listed alphabetically. To see the same methods grouped according to how they are used, please refer to the Categorized List of Methods.

DataHubConnector

`DataHubConnector` — creates a DataHubConnector object.

Syntax

For C++:

```
CDataHubConnector (
   void
   );
```

For Java:

```
DataHubConnector (
   DataHubEventDispatcher parent
   );
```

For C#:

```
DataHubConnector (
   DataHubEventConsumer parent
   );
```

Parameters

`parent`

The parent for this object.

Description

Creates a `DataHubConnector` object.

See Also

`DataHubConnector` Class, Categorized List of Methods

~DataHubConnector

`~DataHubConnector` — destroys a DataHubConnector object.

Syntax

For C++:

```
~DataHubConnector (
   void
   );
```

For Java, and C#:

None.

Description

Destroys a `DataHubConnector` object.

See Also

`DataHubConnector` Class, Categorized List of Methods

activeHeartbeatTimers

`activeHeartbeatTimers` — determines if both heartbeat timers are active.

Syntax

For C++ and C#:

```
bool activeHeartbeatTimers (
   void
   );
```

For Java:

```
boolean activeHeartbeatTimers (
   void
   );
```

Returns

TRUE if both heartbeat timers are active, FALSE otherwise.

Description

This method is used to determine if both heartbeat timers were successfully started. The heartbeat timers are normally started when a connection has been established (see `startHeartbeatTimers`). You may wish to check that the heartbeat mechanism is active before registering a domain or points.

See Also

`startHeartbeatTimers`, `cancelHeartbeatTimers`, `setHeartbeatTimes`

Example

```
    void DataGenerator::onConnectionSuccess (LPCTSTR host, int port)
    {
        _super::onConnectionSuccess (host, port); // starts the heartbeat timers

        // proceed to start heartbeat and specify domain
        int domain_flags = DHC_FLAG_REG_FUTURE|
                           DHC_FLAG_REG_QUALIFY|
                           DHC_FLAG_REG_ONCEONLY;

        if (activeHeartbeatTimers()) // started the heartbeat timers
        {
            if (registerDomain (m_DomainName, (DHC_tRegFlags)domain_flags) == ST_OK)
            {
                RegisterPoints();
            }
        }
    }
```

addPointValue

`addPointValue` — adds a specified amount to a DataHub point value.

Syntax

For C++:

```
ST_STATUS addPointValue (
   CDataHubPoint& point,
   double value
);


ST_STATUS addPointValue (
   LPCTSTR pointname,
   double value
);
```

For Java and C#:

```
Exception addPointValue (
   DataHubPoint point,
   double value
);


Exception addPointValue (
   String pointname,
   double value
);
```

Parameters

point

A `DataHubPoint` object. The name, seconds and nanoseconds members must be valid.

pointname

The name of the point. The point timestamp is automatically set to the current time.

value

The value to add to the current point value.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

Adds the specified value to the current value of the DataHub point. If the DataHub point is not of a numeric type, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_WRONG_TYPE`
msg: `"Wrong type"`

If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_NO_POINT`
msg: `"Point does not exist"`

Examples

```
addPointValue(_T("intPoint1"), 1.0);
```

```
CDataHubPoint point;
point.name = "realPoint2";
setPointTimeStamp (&point);
addPointValue(&point, 1.234);
```

See Also

`appendPointValue`, `multiplyPointValue`, `dividePointValue`, `writePoint`

appendPointValue

`appendPointValue` — appends a string to a DataHub point.

Syntax

For C++:

```
ST_STATUS appendPointValue (
   CDataHub& point,
   LPCTSTR str
);


ST_STATUS appendPointValue (
   LPCTSTR pointname,
   LPCTSTR str
);
```

For Java and C#:

```
Exception appendPointValue (
   DataHub point,
   String str
);


Exception appendPointValue (
   String pointname,
   String str
);
```

Parameters

point

A pointer structure of type `DataHubPoint`. The name, seconds and nanoseconds members must be valid.

pointname

The name of the point. The point timestamp is automatically set to the current time.

str

A string to append to the current point string value.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method appends the specified string to the current string value of the DataHub point. If the DataHub point is not a string type (`PT_TYPE_STRING`), then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_WRONG_TYPE`
msg: `"Wrong type"`

If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_NO_POINT`
msg: `"Point does not exist"`

Examples

```
appendPointValue(_T("strPoint1"), _T("this"));
```

```
CDataHubPoint point;
point.name = "strPoint1";
setPointTimeStamp (&point);
appendPointValue(&point, _T(" and that"));
```

See Also

`addPointValue`, `multiplyPointValue`, `dividePointValue`, `writePoint`

cancelHeartbeatTimers

`cancelHeartbeatTimers` — cancels the heartbeat and timeout timers.

Syntax

For C++, Java, and C#:

```
void cancelHeartbeatTimers (
   void
   );
```

Description

See `setHeartbeatTimes` for more details on the timeout timer feature.

See Also

`setHeartbeatTimes`, `startHeartbeatTimers`

cancelReconnectionTimer

`cancelReconnectionTimer` — stops the delay timer for reconnection attempts.

Syntax

For C++, Java, and C#:

```
void cancelReconnectionTimer (
   void
   );
```

Description

This method cancels the reconnection delay timer. The default behaviour of the `onConnectionSuccess` method is to make a call to `cancelReconnectionTimer`. See `setReconnectionDelay` for more details.

`closeConnection`

`closeConnection` — closes the connection to the DataHub.

Syntax

For C++:

```
void closeConnection (
   void
   );
```

For Java and C#:

```
void closeConnection (
   String reason
   );
```

Description

This method closes the connection to the DataHub.

See Also

`openConnection`, `retryConnection`, `isConnecting`, `isConnected`

createPoint

`createPoint` — creates a DataHub point.

Syntax

For C++:

```
ST_STATUS createPoint (
   CDataHubPoint& point
   );


ST_STATUS createPoint (
   LPCTSTR pointname
   );
```

For Java and C#:

```
Exception createPoint (
   DataHubPoint point
   );


Exception createPoint (
   String pointname
   );
```

Parameters

point

A `DataHubPoint` object. The name member must be valid.

pointname

The name of the point.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method creates a DataHub point. The point quality is set to `PT_QUALITY_BAD` and the timestamp is set to `0`. The point type and value are undefined.

See Also

`registerPoint`, `unregisterPoint`

dividePointValue

`CDataHubConnector: dividePointValue` — divides the value of the named point by the specified value.

Syntax

For C++:

```
ST_STATUS dividePointValue (
   CDataHubPoint& point,
   double value
);


ST_STATUS dividePointValue (
   LPCTSTR pointname,
   double value
);
```

For Java and C#:

```
Exception dividePointValue (
   DataHubPoint point,
   double value
);


Exception dividePointValue (
   String pointname,
   double value
);
```

Parameters

point

A `DataHubPoint` object. The name, seconds and nanoseconds members must be valid.

pointname

The name of the point. The point timestamp is automatically set to the current time.

value

The value by which to divide the current point value.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method divides the current value of the DataHub point by the specified value. If the DataHub point is not of a numeric type, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_WRONG_TYPE`
msg: `"Wrong type"`

If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_NO_POINT`
msg: `"Point does not exist"`

Examples

```
dividePointValue(_T("intPoint1"), 1.0);
```

```
CDataHubPoint point;
point.name = "realPoint2";
setPointTimeStamp (&point);
dividePointValue(&point, 1.234);
```

See Also

`appendPointValue`, `addPointValue`, `multiplyPointValue`, `writePoint`

escapedString

`escapedString` — prepares a string for use with `writeCommand` (Java and C# only).

Syntax

For Java:

```
String escapedString (
   String str,
   boolean quoted
);


String escapedString (
   String str,
   boolean quoted,
   boolean special_only
);
```

For C#:

```
String escapedString (
   String str,
   bool quoted
);


String escapedString (
   String str,
   bool quoted,
   bool special_only
);
```

Parameters

This has not yet been documented.

Returns

This has not yet been documented.

Description

This method prepares a string for use with `writeCommand`.

See Also

`writeCommand`

getCnxState

`getCnxState` — retrieves the operational state of the connector object.

Syntax

For C++, Java, and C#:

```
DHC_tState getCnxState (
   void
   );
```

Returns

The current state of the connector object, a member of `DHC_tState` as defined below.

Description

Retrieves the state of the DataHub connector object. The state is primarily informational, since the user cannot directly change it.

State	Description
`DHC_STATE_NONE`	This is the initial (`0`) state, and is immediately changed to `DHC_STATE_IDLE` within the constructor unless a fundamental initialization error has occurred (eg. Window creation).
`DHC_STATE_IDLE`	A connection has never been attempted.
`DHC_STATE_INITIALIZED`	Internal initialization completed (by the first attempt to connect) and ready to connect.
`DHC_STATE_CONNECTING`	Currently in the process of connecting. See `getCnxSubStateString` for detailed connection sub-state.
`DHC_STATE_CONNECTING_CLOSING`	This is a transient internal state used to force closure of an open connection before attempting to reconnect.
`DHC_STATE_CONNECTED`	A DataHub connection exists.
`DHC_STATE_ERROR`	An error was detected. See `getErrString`. This is generally a transient state, since the default behaviour is to attempt to reconnect.
`DHC_STATE_RETRY_DELAY`	The reconnection timer is active and after which an attempt will be made to reconnect (see `setReconnectionDelay`).

See Also

`onStatusChange`, `getCnxStateString`, `getCnxSubStateString`, `isConnecting`, `isConnected`, `getErrString`, `getReconnectionDelay`, `setReconnectionDelay`

getCnxStateString

`getCnxStateString` — retrieves a string corresponding to the operational state.

Syntax

For C++:

```
CString getCnxStateString (
   void
   );
```

For Java and C#:

```
String getCnxStateString (
   void
   );
```

Returns

The string corresponding to the operational state of the object (as returned by `getCnxState`).

Description

This method retrieves a string corresponding to the operational state.

See Also

`getCnxState`

getCnxSubStateString

`getCnxSubStateString` — provides detailed information on the connection state (C++ only).

Syntax

For C++:

```
CString getCnxSubStateString (
   void
   );
```

Returns

A string containing a short description of the connection status.

Description

This method provides some additional insight into the status of the DataHub connection. A new status string may be obtained on each `onStatusChange`, along with the main `getCnxStateString`, and used for logging or otherwise indicating the connection sequence.

See Also

`onStatusChange` `getCnxStateString`, `getCnxState`

getDefaultDomain

`getDefaultDomain` — returns the current default domain.

Syntax

For C++:

```
LPCTSTR getDefaultDomain (
   void
   );
```

For Java, and C#:

```
String getDefaultDomain (
   void
   );
```

Returns

The domain name string, or `NULL`[1] if none has been set.

Description

This method accesses the domain name string specified by the `setDefaultDomain` method. The value returned is based on a local (client) copy, and is therefore not dependent on the connection status.

See Also

`setDefaultDomain`

Notes

1. `null in Java and C#.`

`getErrString`

`getErrString` — retrieves the last error string (C++ only).

Syntax

For C++:

```
LPCTSTR getErrString  (
   void
   );
```

Returns

The string corresponding to the last error detected.

See Also

`onError`

getHeartbeat

`getHeartbeat` — retrieves the heartbeat timer period.

Syntax

For C++, Java, and C#:

```
long getHeartbeat (
   void
   );
```

Returns

The period in milliseconds of the heartbeat timer.

Description

See `setHeartbeatTimes` for more details on the heartbeat timer feature.

See Also

`setHeartbeatTimes`, `getTimeout`

getHostName

`getHostName` — retrieves the host name connection parameter.

Syntax

For C++:

```
CString getHostName (
   void
   );
```

For Java and C#:

```
String getHostName (
   void
   );
```

Returns

A string containing the currently set host name.

Description

This method retrieves the host name string set by the `setConnectionParms` command, or an empty string if it has never been set. The value is not affected by `openConnection` or the status of the connection.

getPort

`getPort` — retrieves the port connection parameter.

Syntax

For C++, Java, and C#:

```
int getPort (
   void
   );
```

Returns

The connection port number.

Description

This method retrieves the port number set by the `setConnectionParms` command, or `0` if it has never been set. If `setConnectionParms` was used to specify a service name, then the port will be the result of an attempt to convert or look up the servicename specified. The value is not affected by `openConnection` or the status of the connection.

getReconnectionDelay

`getReconnectionDelay` — retrieves the delay time between reconnection attempts.

Syntax

For C++, Java, and C#:

```
long getReconnectionDelay (
   void
   );
```

Returns

The reconnection delay time setting, in milliseconds.

Description

See `setReconnectionDelay` for more details.

getServiceName

`getServiceName` — retrieves the port service name connection parameter (C++ only).

Syntax

For C++:

```
CString getServiceName (
   void
   );
```

Returns

A string containing the currently set port service name.

Description

This method retrieves the port service name string set by the `setConnectionParms` command, or an empty string if it has never been set. If `setConnectionParms` was used to set the port directly (as an integer), then the servicename will not have been set. The value is not affected by `openConnection` or the status of the connection.

getTimeout

`getTimeout` — retrieves the timeout timer period.

Syntax

For C++, Java, and C#:

```
long getTimeout (
   void
   );
```

Returns

The period in milliseconds of the timeout timer.

Description

See `setHeartbeatTimes` for more details on the timeout timer feature.

See Also

`setHeartbeatTimes`, `getHeartbeat`

initializePointCache

`initializePointCache` — initializes local point cache usage.

Syntax

For C++:

```
ST_STATUS initializePointCache (
   void
   );
```

For Java, and C#:

```
Exception initializePointCache (
   void
   );
```

Returns

For C++:

`ST_OK` if the point cache was successfully initialized, otherwise `ST_ERROR`.

Description

The point cache is a list of all points received. It is automatically built and maintained as each point update is received from the DataHub. In order for a point to exist in the cache, a `readPoint`, `registerPoint`, or `registerDomain` was required to generate the point update. The value, type, and timestamp of the cached points is updated, but the point itself is persistent.

The point cache allows the user to associate data with the point (through its `m_userdata` member) and quickly access it when the point is updated (from `onPointChange`). The point cache also provides a way to synchronously read a current point value, avoiding the `readPoint` method which provides the point value asynchronously.

Example 1

This example illustrates use of the point cache to efficiently update a user interface control with the latest point value.

```
....
// during initialization, we register for all points
registerDomain("myDomain",
                DHC_FLAG_REG_FUTURE | DHC_FLAG_ONCEONLY_FUTURE);
....

// associate an MFC control with a point
//  perhaps this is called on user entering a desired point name
//  (don't forget to clear the previously associated point userdata!)
void onUiPointNameChanged(LPCTSTR pointname, CStatic *pValueWnd)
{
    DataHubPoint *ppoint = lookupPoint (pointname);
    if (ppoint)
    {
        ppoint->m_userdata = pointname;
    }
}
void onPointChange (DataHubPoint point)
{
```

```
    if (point.m_userdata)
    {
        // update the MFC control
        CStatic *pTxt = point.m_userdata;
        CString str;
        str.Format("%f", point.getDoubleValue());
        pTxt->SetWindowText(str);
    }
}
```

Example 2

This example illustrates use of the point cache to provide immediate access to a point value.

```
....
// part of the initialization code
registerDomain("myDomain",
               DHC_FLAG_REG_FUTURE | DHC_FLAG_ONCEONLY_FUTURE);
....

// read point value:
DataHubPoint *point = lookupPoint("IntPoint1");
int ival = point->getIntValue();
....
```

See Also

`registerPoint`, `unregisterPoint`, `createPoint`

isConnected

`isConnected` — indicates whether a connection has been established.

Syntax

For C++ and C#:

```
bool isConnected (
   void
   );
```

For Java:

```
boolean isConnected (
   void
   );
```

Returns

TRUE is a connection to the DataHub has been established, FALSE otherwise.

See Also

`isConnecting`, `openConnection`, `retryConnection`, `closeConnection`

isConnecting

`isConnecting` — indicates whether a connection attempt is in progress.

Syntax

For C++ and C#:

```
bool isConnecting (
   void
   );
```

For Java:

```
boolean isConnecting (
   void
   );
```

Returns

TRUE if the object is currently in the process of establishing a connection with the DataHub, FALSE otherwise.

See Also

`isConnected`, `openConnection`, `retryConnection`, `closeConnection`

lookupPoint

`lookupPoint` — accesses a point from the point cache.

Syntax

For C++:

```
DataHubPoint* lookupPoint (
   LPCTSTR pointname
   );
```

For Java, and C#:

```
DataHubPoint lookupPoint (
   String pointname
   );
```

Parameters

`pointname`

The name of the point.

Returns

The corresponding point object in the point cache.

Description

This method returns a pointer to a cached `DataHubPoint` object matching the specified name object. If no object matching the name is in the cache, then `NULL`[1] is returned.

See Also

`initializePointCache`

Notes

1. `null` in Java and C#.

multiplyPointValue

`multiplyPointValue` — multiplies a DataHub point value by a specified amount.

Syntax

For C++:

```
ST_STATUS multiplyPointValue (
   CDataHubPoint& point,
   double value
);


ST_STATUS multiplyPointValue (
   LPCTSTR pointname,
   double value
);
```

For Java and C#:

```
Exception multiplyPointValue (
   DataHubPoint point,
   double value
);


Exception multiplyPointValue (
   String pointname,
   double value
);
```

Parameters

point

A `DataHubPoint` object. The name, seconds and nanoseconds members must be valid.

pointname

The name of the point. The point timestamp is automatically set to the current time.

value

The value by which to multiply the current point value.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.

- ST_ERROR if the connection socket is unable to send the message.

Description

This method multiplies the current value of the DataHub point by the specified value. If the DataHub point is not of a numeric type, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_WRONG_TYPE
msg: "Wrong type"

If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_NO_POINT
msg: "Point does not exist"

Examples

```
multiplyPointValue(_T("intPoint1"), 1.0);
```

```
CDataHubPoint point;
point.name = "realPoint2";
setPointTimeStamp (&point);
multiplyPointValue(&point, 1.234);
```

See Also

appendPointValue, addPointValue, dividePointValue, writePoint

openConnection

`openConnection` — attempts to establish a connection to the DataHub.

Syntax

For C++, Java, and C#:

```
void openConnection (
   void
   );
```

Description

This method attempts to establish a connection to the DataHub, using the parameters set by `setConnectionParms`.

See Also

`retryConnection`, `closeConnection`, `isConnecting`, `isConnected`

readPoint

`readPoint` — gets the value of a DataHub point.

Syntax

For C++:

```
ST_STATUS readPoint (
   CDataHubPoint& point,
   bool create=TRUE,
   bool force=TRUE
);


ST_STATUS readPoint (
   LPCTSTR pointname,
   bool create=TRUE,
   bool force=TRUE
);
```

For Java:

```
Exception readPoint (
   DataHubPoint point
   );


Exception readPoint (
   DataHubPoint point,
   boolean create,
   boolean force
);


Exception readPoint (
   String pointname
   );


Exception readPoint (
   String pointname,
   boolean create,
   boolean force
);
```

For C#:

```
Exception readPoint (
   DataHubPoint point
   );


Exception readPoint (
   DataHubPoint point,
   bool create,
   bool force
);


Exception readPoint (
```

```
   String pointname
   );


Exception readPoint (
   String pointname,
   bool create,
   bool force
);
```

Parameters

point

A DataHubPoint object. The name member must be valid.

pointname

The name of the point.

create

If TRUE, then the point is created if it does not already exist. If FALSE and the point does not exist, then an error message will be generated.

force

If a valid connection to the DataHub exists but the message is undeliverable immediately (due to the TCP buffer being full), then if the force parameter is TRUE, the message is queued and will be transmitted when possible.

Returns

For C++:

- ST_OK if the command was successfully sent to the DataHub.
- ST_NO_TASK if a connection to the DataHub does not exist.
- ST_ERROR if the connection socket is unable to send the message.

Description

This method requests that the value of the specified point be transmitted. The point value is received as a point message, which causes the virtual method onPointChange to be called. The point value cannot be obtained synchronously.

If the *create* parameter is FALSE and the point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_NO_POINT
msg: "Point does not exist"

Examples

```
readPointValue(_T("strPoint1"));
CDataHubPoint point;
point.name = "strPoint1";
readPoint(&point, FALSE);
```

See Also

writePoint

registerDomain

`registerDomain` — registers to receive updates from domain points.

Syntax

For C++:

```
ST_STATUS registerDomain (
   LPCTSTR domainname,
   int flags
);
```

For Java and C#:

```
Exception registerDomain (
   String domainname,
   int flags
);
```

Parameters

`domainname`

The name of a domain.

`flags`

Specifies conditions and actions associated with registering the domain points, as follows (flags may be combined):

- `DHC_FLAG_REG_FUTURE`: points created in the future are also affected. If not set, then only those points existing (in the domain) at the time the command message is received by the DataHub are registered.
- `DHC_FLAG_REG_QUALIFY`: point names are to be transmitted with the domain name prepended to the point name, as *`domainname`*`:`*`pointname`*.
- `DHC_FLAG_REG_ONCEONLY`: the point values are transmitted only once, and the points remain unregistered. This mode is useful for obtaining an initial list of available points from which a selection is made of which ones to register.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method registers the client to receive updates from the DataHub when the value of any points in the specified domain change. The `onPointChange` method is called for each point update received. The `registerDomain` method may be called more than once, and on different domains.

Upon registering the domain, all points currently in that domain of the DataHub are immediately transmitted to the client. Points that are subsequently created will only be sent if the `DHC_FLAG_REG_FUTURE` flag is specified. The `DHC_FLAG_REG_QUALIFY` flag is useful for distinguishing points from different domains that have the same name.

In a typical scenario, either there is no need to register for all points, or the point names are not even known. In either case, you can use the `DHC_FLAG_REG_ONCEONLY` flag to generate an initial list of all available points in the specified domain. Then you can call `registerPoint` on any points of interest, as they are received in `onPointChange`.

See Also

`unregisterPoint`, `setDefaultDomain`, `registerPoint`

registerPoint

`registerPoint` — registers to receive updates from a DataHub point.

Syntax

For C++:

```
ST_STATUS registerPoint (
   CDataHubPoint& point,
   bool create=TRUE
);


ST_STATUS registerPoint (
   LPCTSTR pointname,
   bool create=TRUE
);
```

For Java:

```
Exception registerPoint (
   DataHubPoint point,
   boolean create
);


Exception registerPoint (
   String pointname,
   boolean create
);
```

For C#:

```
Exception registerPoint (
   DataHubPoint point,
   bool create
);


Exception registerPoint (
   String pointname,
   bool create
);
```

Parameters

point

A `DataHubPoint` object. The name member must be valid.

pointname

The name of the point.

create

If TRUE, then the point is created if it does not already exist. If FALSE and the point does not exist, then an error message will be generated.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

When a point is registered with the DataHub, any changes to the value of that point in the DataHub will cause the `onPointChange` method to be called. If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_NO_POINT`
msg: `"Point does not exist"`

See Also

`writePoint`, `unregisterPoint`, `createPoint`, `registerDomain`

retryConnection

`retryConnection` — opens a new connection to the DataHub (C++ only).

Syntax

For C++:

```
void retryConnection (
   void
   );
```

Description

This method attempts to reestablish a connection to the DataHub.

See Also

`openConnection`, `closeConnection`, `isConnecting`, `isConnected`

sendBinaryPointMessages

`sendBinaryPointMessages` — formats data in binary mode (C++ only).

Syntax

For C++:

```
ST_STATUS sendBinaryPointMessages (
   bool enable
   );
```

Parameters

enable

`TRUE` enables binary mode, `FALSE` disables it.

Returns

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if no connection to the DataHub is established.
- `ST_TOO_LARGE` if the message string exceeds the message buffer size.
- `ST_ERROR` if the format contains an error or if the connection socket is unable to send the message.

Description

This method tells the `CDataHubConnector` instance to format data in binary mode, and also to request binary mode transmissions from the DataHub. Binary messages are more CPU efficient than ASCII messages.

sendLispMessage

`sendLispMessage` — sends a message to the DataHub (C++ only).

Syntax

For C++:

```
ST_STATUS sendLispMessage (
   bool   force,
   char* format,
    ...
);
```

Parameters

force

If a valid connection to the DataHub exists but the message is undeliverable immediately (due to the TCP buffer being full), then if the force parameter is `TRUE`, the message is queued and will be transmitted when possible.

format

The text formatting string (see below).

...

Parameters required by the specified format.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if no connection to the DataHub is established.
- `ST_TOO_LARGE` if the message string exceeds the message buffer size.
- `ST_ERROR` if the format contains an error or if the connection socket is unable to send the message.

Description

This method is used to format and send commands to the DataHub, and should only be used by those very familiar with the operation of the DataHub. The DataHub command set is described in the Using Commands chapter of the Cogent DataHub manual. To send a command to the DataHub, it must be formatted such that strings and control characters are preserved through the message transfer and delivery process. The specialized format control specifiers of sendLispMessage make this easy to do.

The `writeCommand` method gives a similar functionality for Java and C#.

The format control string is similar to that used by `printf`, with the following differences:

- The `%A` field type specifier escapes all occurences of double quotes, i.e., substitutes `\"` for each occurence of `"`.

- The %a field type specifier escapes all occurences of the following characters in addition to the double quote: \ CR LF FF TAB ( ).
- The %s field type specifier escapes the same characters as %a, and also encloses the string in double quotes.
- The %d (or %i) field type specifier assumes a parameter of type int, or of type long if preceeded by an 'l' type specifier (%ld). No other type length specifiers are supported.
- The %f and %g field type specifiers assume a parameter of type double. No type length specifiers are supported.
- Other field type specifiers such as %c, %p, %n, %o, %u, %x and %e are not supported.
- The '*' field width specifier is not supported.

The string parameters corresponding to the %s, %a or %A field type specifiers *must* be of type char *.

Example

```
void CSetpoint::BuildObjectHierarchy(LPCTSTR sDomain, LPCTSTR sAssembly)
{
    CT2A aDomain(sDomain);
    char *domain = aDomain;
    CT2A aAssembly(sAssembly);
    char *assembly = aAssembly;
    CT2A aName(sName);
    char *name = aName;

    // (assembly domain name)
    pCnx->sendLispMessage(TRUE, "(assembly %s CSetpoint)", domain);
    // (subassembly domain assemblyname subassemblyname instancename)
    pCnx->sendLispMessage(TRUE, "(subassembly %s %s CSetpoint %s)", domain, assembly, name);
    // (property domain attrname propid propname type rw dflt_value dflt_conf)
    pCnx->sendLispMessage(TRUE, "(property %s CSetpoint auto AutoMode int rw 0 100)", domain);
    pCnx->sendLispMessage(TRUE, "(property %s CSetpoint auto AutoTime r8 rw 0 100)", domain);
}
```

sendLogin

`sendLogin` — transmits the user name and password.

Syntax

For C++:

```
void sendLogin (
   void
   );
```

For Java and C#:

```
public void sendLogin (
   void
   );
```

Description

This method transmits the user name and password previously set by a call to `setUsername` or `setConnectionParms`. The `sendLogin` method is normally called automatically when a successful connection is made to the DataHub, prior to the `onConnectionSuccess` user callback.

setConnectionParms

`setConnectionParms` — sets the connection parameters.

Syntax

For C++:

```
void setConnectionParms (
   LPCTSTR hostname,
   LPCTSTR servicename
);


void setConnectionParms (
   LPCTSTR hostname,
   int port
);


void setConnectionParms (
   LPCTSTR hostname,
   int port,
   LPCTSTR username,
   LPCTSTR password
);
```

For Java and C#:

```
void setConnectionParms (
   String hostname,
   int port
);


void setConnectionParms (
   String hostname,
   int port,
   String username,
   String password
);
```

Parameters

`hostname`

The name of the host running the DataHub.

`servicename`

The name of the port on which to connect.

`port`

The connection port number.

`username`

The name of a user.

`password`

A password for that user.

Description

This method sets the connection parameters to be used by the `openConnection` method. The port may be specified as either a string or directly by its number. The string may represent the port number, which is simply converted to an integer, or the symbolic port servicename. *If* a servicename is specified, then a lookup is performed immediately, and the resulting port number may be verified by following with a call to `getPort`.

The expanded syntax allows for setting the user name and password. See `setUsername` for more details.

setDefaultDomain

`setDefaultDomain` — sets the default domain.

Syntax

For C++:

```
ST_STATUS setDefaultDomain (
   LPCTSTR domainname
   );
```

For Java and C#:

```
Exception setDefaultDomain (
   String domainname
   );
```

Parameters

`domainname`

The name of the domain.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method sets the name of the DataHub domain to be automatically associated with all unqualified point names (i.e., point names that do not specify a domain). Since in many applications there is a single domain to which most if not all points belong, this feature makes point references simpler and shorter. Points in other domains can still be referenced by qualifying the point name with a specific domain name (see `qualifyName`).

If the DataHub is not connected at the time this method is called, then the domain will be transmitted when the connection is established.

See Also

`getDefaultDomain`, `qualifyName`

setHeartbeatTimes

`setHeartbeatTimes` — sets the period of the heartbeat timers.

Syntax

For C++:

```
ST_STATUS setHeartbeatTimes (
   long heartbeat_ms,
   long timeout_ms
);
```

For Java and C#:

```
Exception setHeartbeatTimes (
   long heartbeat_ms,
   long timeout_ms
);
```

Parameters

`heartbeat_ms`

The period in milliseconds for the heartbeat timer, or `0` if a timer will not be used.

`timeout_ms`

The period in milliseconds for the timeout timer, or `0` if a timer will not be used.

Returns

For C++:

`ST_OK` if the timer values accepted, otherwise `ST_ERROR`. `ST_ERROR` can occur for the following reasons:

- The timeout timer value is not greater than heartbeat (if both values are greater than `0`).
- The timers were already active and either timer failed to restart.

Description

The DataHub and API provide two heartbeat services to ensure the integrity of the connection:

1. A local, internal timeout timer, which when it fires, will close the connection and generate an error.
2. A periodic 'alive' heartbeat message from the DataHub, which triggers the virtual method `onAlive`.

The timers can be started with `startHeartbeatTimers` once the respective periods have been set. If a timer is already running when `setHeartbeatTimes` is called, then that timer is restarted with the new period. If the timer is already running and the specified period is `0`, then the timer is stopped.

The timeout timer is automatically reset whenever there is any activity over the connection, including the heartbeat timer message.

See Also

`getHeartbeat`, `getTimeout`, `startHeartbeatTimers`, `cancelHeartbeatTimers`

setPointLock

`setPointLock` — sets the lock attributes of a DataHub point.

Syntax

For C++:

```
ST_STATUS setPointLock (
   CDataHubPoint& point,
   bool locked
);


ST_STATUS setPointLock (
   LPCTSTR pointname,
   bool locked
);
```

For Java:

```
Exception setPointLock (
   DataHubPoint point,
   boolean locked
);


Exception setPointLock (
   String pointname,
   boolean locked
);
```

For C#:

```
Exception setPointLock (
   DataHubPoint point,
   bool locked
);


Exception setPointLock (
   String pointname,
   bool locked
);
```

Parameters

point

A `DataHubPoint` object. The name and security members must be valid.

pointname

The name of the point.

locked

If `TRUE`, the point will have its locked attribute set, otherwise the locked attribute is cleared.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method sets the lock attributes of a DataHub point.

See Also

`setPointSecurity`

setPointSecurity

`setPointSecurity` — sets the security attributes of a DataHub point.

Syntax

For C++:

```
ST_STATUS setPointSecurity (
   CDataHubPoint& point,
   int security
);


ST_STATUS setPointSecurity (
   LPCTSTR pointname,
   int security
);
```

For Java and C#:

```
Exception setPointSecurity (
   CDataHubPoint& point,
   int security
);


Exception setPointSecurity (
   String pointname,
   int security
);
```

Parameters

point

A `DataHubPoint` object. The name and security members must be valid.

pointname

The name of the point.

security

The new security level to which the DataHub point will be set.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method sets the security attributes of a DataHub point. An update to a DataHub point will be rejected unless the security level associated with the new point value is greater than or equal to the corresponding DataHub point. The default DataHub point security level is `0`.

See Also

`setPointLock`

setReconnectionDelay

`setReconnectionDelay` — sets the delay time between reconnection attempts.

Syntax

For C++, Java, and C#:

```
void setReconnectionDelay (
   long recon_ms
   );
```

Parameters

`recon_ms`

The delay time, in milliseconds, after connection failure before another attempt is automatically made.

Description

The `DataHubConnector` class provides an internal facility to automatically attempt reconnection in event of a failure. The default `onConnectionFailure` method will trigger the reconnection timer.

When the timer fires, the `retryConnection` method is called to cancel the timer and attempt to connect.

See Also

`getReconnectionDelay`, `startReconnectionTimer`, `cancelReconnectionTimer`

setUsername

`setUsername` — stores a user name and password for this connection.

Syntax

For C++:

```
void setUsername (
   LPCTSTR username,
   LPCTSTR password
);
```

For Java and C#:

```
public void setUsername (
   String username,
   String password
);
```

Parameters

`username`

The name of a user.

`password`

A password for that user.

Description

This method stores a user name and password for this connection, to be transmitted by a subsequent call to `sendLogin`. When a subsequent successful connection is made to the DataHub, `sendLogin` will be called prior to the `onConnectionSuccess` callback. The .Net and Java implementations of `setUsername` will disconnect from the DataHub if the connection is currently active. The C++ implementation will not. If either of the username or password parameters is `""` or `NULL`, then no username or password will be sent on the next successful connection.

See Also

`sendLogin`, `setConnectionParms`

shutdown

`shutdown` — prepares for an application shutdown or disconnect (Java and C# only).

Syntax

For Java, and C#:

```
void shutdown (
   String reason
   );
```

Parameters

`reason`

A character string that may be passed to the `onConnectionFailure` callback indicating the reason for the shutdown. If *`reason`* is `null`, a default string will be used.

Description

This method cleans up all resources, timers and sockets associated with the connection in preparation for application shutdown or removal of the `DataHubConnector` object. The application should not call any methods of the object or access any of its members after this method call is made. The result of any access to the object after this call is undefined.

See Also

`onConnectionFailure`

startHeartbeatTimers

`startHeartbeatTimers` — starts the heartbeat and timeout timers.

Syntax

For C++:

```
ST_STATUS startHeartbeatTimers (
   void
   );
```

For Java and C#:

```
Exception startHeartbeatTimers (
   void
   );
```

Returns

For C++:

`ST_OK` if the timers were successfully started, otherwise `ST_ERROR`.

Description

This method is used to start the two heartbeat timers. If a timer period has been set to `0`, then that heartbeat timer function is disabled. The default behaviour of the `onConnectionSuccess` method is to make a call to `startHeartbeatTimers`.

See Also

`setHeartbeatTimes`, `cancelHeartbeatTimers`, `onConnectionSuccess`

startReconnectionTimer

`startReconnectionTimer` — starts the delay timer for reconnection attempts.

Syntax

For C++, Java, and C#:

```
void startReconnectionTimer (
   void
   );
```

Description

This method is used to start the reconnection timer. The default behaviour of the `onConnectionFailure` method is to make a call to `startReconnectionTimer`. See `setReconnectionDelay` for more details.

unregisterPoint

`unregisterPoint` — stops receiving updates from a DataHub point.

Syntax

For C++:

```
ST_STATUS unregisterPoint (
   CDataHubPoint& point
   );


ST_STATUS unregisterPoint (
   LPCTSTR pointname
   );
```

For Java and C#:

```
Exception unregisterPoint (
   DataHubPoint point
   );


Exception unregisterPoint (
   String pointname
   );
```

Parameters

`point`

A `DataHubPoint` object. The name member must be valid.

`pointname`

The name of the point.

Returns

For C++:

- `ST_OK` if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the `onSuccess` or `onError` message handlers.
- `ST_NO_TASK` if a connection to the DataHub does not exist.
- `ST_ERROR` if the connection socket is unable to send the message.

Description

This method stops changes to the specified DataHub point from invoking the `onPointChange` method. If the DataHub point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: `ST_NO_POINT`
msg: `"Point does not exist"`

See Also

writePoint, registerPoint

writeCommand

`writeCommand` — sends a command to the DataHub (Java and C# only).

Syntax

For Java and C#:

```
Exception writeCommand (
   String command
   );
```

Parameters

command

A concatenation of `escapedString`s whose original format was a Lisp command.

Returns

This has not yet been documented.

Description

This method is used to send formatted commands to the DataHub, and should only be used by those very familiar with the operation of the DataHub. The DataHub command set is described in the Using Commands chapter of the Cogent DataHub manual. To send a command to the DataHub, it must be formatted such that strings and control characters are preserved through the message transfer and delivery process. You must concatenate together the necessary command string, using the `escapedString` method to provide the necessary protection of strings.

The `sendLispMessage` method gives a similar functionality for C++.

See Also

`escapedString`

writePoint

`writePoint` — writes a new value to a DataHub point.

Syntax

For C++:

```
ST_STATUS writePoint (
   CDataHubPoint& point,
   bool create=TRUE,
   int force=FALSE
);


ST_STATUS writePoint (
   LPCTSTR pointname,
   int value,
   bool create=TRUE,
   bool force=FALSE
);


ST_STATUS writePoint (
   LPCTSTR pointname,
   double value,
   bool create=TRUE,
   bool force=FALSE
);


ST_STATUS writePoint (
   LPCTSTR pointname,
   LPCTSTR value,
   bool create=TRUE,
   bool force=FALSE
);
```

For Java:

```
Exception writePoint (
   DataHubPoint point,
   boolean create,
   boolean force
);


Exception writePoint (
   String pointname,
   String value,
   boolean create,
   boolean force
);


Exception writePoint (
   String pointname,
   String value
);


Exception writePoint (
   String pointname,
```

```
   double value,
   boolean create,
   boolean force
);


Exception writePoint (
   String pointname,
   double value
);


Exception writePoint (
   String pointname,
   int value,
   boolean create,
   boolean force
);


Exception writePoint (
   String pointname,
   int value
);


Exception writePoint (
   String pointname,
   int type,
   String value_as_string,
   boolean create,
   boolean force
);
```

For C#:

```
Exception writePoint (
   DataHubPoint point,
   bool create,
   bool force
);


Exception writePoint (
   String pointname,
   String value,
   bool create,
   bool force
);


Exception writePoint (
   String pointname,
   String value
);


Exception writePoint (
   String pointname,
   double value,
   bool create,
   bool force
);
```

```
Exception writePoint (
   String pointname,
   double value
);


Exception writePoint (
   String pointname,
   int value,
   bool create,
   bool force
);


Exception writePoint (
   String pointname,
   int value
);


Exception writePoint (
   String pointname,
   int type,
   String value_as_string,
   bool create,
   bool force
);
```

Parameters

point

A DataHubPoint object. The name, type, value, seconds and nanoseconds members must be valid.

create

If TRUE, then the point is created if it does not already exist. If FALSE and the point does not exist, then an error message will be generated.

force

If a valid connection to the DataHub exists but the message is undeliverable immediately (due to the TCP buffer being full), then if the force parameter is TRUE, the message is queued and will be transmitted when possible.

pointname

The name of the point. The point timestamp is automatically set to the current time.

value

The integer, double, or string value to write to the point. The type of the point in the DataHub will be changed accordingly.

Returns

For C++:

- ST_OK if the command was successfully sent to the DataHub. Since the command is sent asynchronously, the actual success or failure of the command must be determined through the onSuccess or onError message handlers.
- ST_NO_TASK if a connection to the DataHub does not exist.
- ST_ERROR if the connection socket is unable to send the message.

Description

This method writes the given point to the DataHub. If a DataHubPoint is used, then the point name, type, value, quality and timestamp members must be set. If the point is specified by name, then the type is determined by the specific overloaded method and the timestamp is set to the current time.

If a domain name has been associated with the client (see setDefaultDomain), then the point will be written to that domain, otherwise it will be written to the domain named default. In any case, the domain can always be overridden by qualifying the point name as *domain name*:*point name* (see qualifyName).

If the point has been registered (see registerPoint), and the DataHub point is successfully written, then the onPointEcho method will be invoked.

If the *create* parameter is FALSE and the point does not exist, then the DataHub will respond with an error, and onError will be called with the following arguments:

status: ST_NO_POINT
msg: "Point does not exist"

Examples

```
CDataHubPoint point;
point.name = "point1";
point.type = PT_TYPE_REAL;
point.quality = PT_QUALITY_GOOD;
point.value = 123.456;
setPointTimeStamp(&point);
writePoint(&point, FALSE); // point must exist or error is generated
// write real-valued point, create if needed
writePoint(_T("point2"), 123.456);
```

See Also

readPoint

II. Callback Methods

Table of Contents

These are the callback methods associated with the `DataHubConnector` class. For C++, they are methods of the `DataHubConnector` class itself. For Java and C#, they are methods of the `DataHubEventConsumer` interface.

onAlive

`onAlive` — a virtual method invoked on receipt of a heartbeat from the DataHub.

Syntax

For C++, Java, and C#:

```
void onAlive (
   void
   );
```

Description

This method is invoked upon receipt of the 'alive' hearbeat message from the DataHub. See `setHeartbeatTimes` for more details on this feature.

See Also

`setHeartbeatTimes`, `getHeartbeat`, `startHeartbeatTimers`, `cancelHeartbeatTimers`

onAsyncMessage

`onAsyncMessage` — a virtual method invoked on receipt of a DataHub message.

Syntax

For C++:

```
virtual void onAsyncMessage (
   int argc,
   char** argv
);
```

For Java and C#:

```
virtual void onAsyncMessage (
   String[] arguments
   );
```

Parameters

`argc`

The number of parameters in the DataHub message.

`argv`, *`arguments`*

An array of strings which are the parameters in the DataHub message.

Description

This virtual method is invoked on receipt of a DataHub message. This method is only called if none of the other callbacks are called.

See Also

`onStatusChange`, `onSuccess`, `onError`, `onConnectionSuccess`, `onConnectionFailure`

onConnectionFailure

`onConnectionFailure` — a virtual method invoked when a connection or attempt to connect fails.

Syntax

For C++:

```
virtual void onConnectionFailure (
   LPCTSTR host,
   int port,
   LPCTSTR reason
);
```

For Java and C#:

```
virtual void onConnectionFailure (
   String host,
   int port,
   String reason
);
```

Parameters

`host`

The name of the host running the DataHub.

`port`

The connection port number.

`reason`

The reason for the failure.

Description

This virtual method is invoked when a connection or attempt to connect fails. The default behaviour is to make a call to `startReconnectionTimer`.

See Also

`onStatusChange`, `onConnectionSuccess`, `startReconnectionTimer`

onConnectionSuccess

`onConnectionSuccess` — a virtual method invoked when a connection is established.

Syntax

For C++:

```
virtual void onConnectionSuccess (
   LPCTSTR host,
   int port
);
```

For Java and C#:

```
virtual void onConnectionSuccess (
   String host,
   int port
);
```

Parameters

`host`

The name of the host running the DataHub.

`port`

The connection port number.

Description

This virtual method is invoked when a connection is established. The default behaviour is to make a call to `cancelReconnectionTimer`.

See Also

`onStatusChange`, `onConnectionFailure`, `cancelReconnectionTimer`

onError

`onError` — a virtual method invoked on receipt of an error message.

Syntax

For C++:

```
virtual void onError  (
   ST_STATUS status,
   LPCTSTR err_str,
   LPCTSTR cmd,
   LPCTSTR parms
);
```

For Java and C#:

```
virtual void onError  (
   int status,
   String err_str,
   String cmd,
   String parms
);
```

Parameters

`status`

The status code returned by the DataHub.

`err_str`

The error string, providing more detailed information about the error.

`cmd`

The original command sent to the DataHub to which this reply corresponds.

`parms`

The list of parameters sent as part of the command, as a single space-separated string, or `NULL`[1] if none are returned. If the command involved multiple parameters, the list may be truncated.

Description

This virtual method is invoked on receipt of an error message.

See Also

`getErrString`, `onSuccess`, `onConnectionSuccess`, `onConnectionFailure`

Notes

1. `null` in Java and C#.

onPointChange

`onPointChange` — a virtual method invoked on receipt of a point value change.

Syntax

For C++:

```
virtual void onPointChange (
   CDataHubPoint& point
   );
```

For Java and C#:

```
virtual void onPointChange (
   DataHubPoint point
   );
```

Parameters

`point`

A `DataHubPoint` object. The name member must be valid.

Description

This virtual method is invoked on receipt of a point value change.

See Also

`onPointEcho`

onPointEcho

`onPointEcho` — a virtual method invoked on receipt of a locally changed point value.

Syntax

For C++:

```
virtual void onPointEcho (
   CDataHubPoint& point
   );
```

For Java and C#:

```
virtual void onPointEcho (
   DataHubPoint point
   );
```

Parameters

`point`

A `DataHubPoint` object. The name member must be valid.

Description

This virtual method is invoked on receipt of a locally changed point value.

See Also

`onPointChange`

onStatusChange

`onStatusChange` — a virtual method invoked on change of status.

Syntax

For C++, Java, and C#:

```
virtual void onStatusChange (
   DHC_tState prev_state,
   DHC_tState state
);
```

Parameters

`prev_state`

The previous state.

`state`

The current state.

Description

This is a virtual method which is invoked on change of status. See `getCnxState` for a definition of the possible states.

See Also

`getCnxState`

onSuccess

`onSuccess` — a virtual method invoked on receipt of a success message.

Syntax

For C++:

```
virtual void onSuccess (
   LPCTSTR cmd,
   LPCTSTR parms
);
```

For Java and C#:

```
virtual void onSuccess (
   String cmd,
   String parms
);
```

Parameters

`cmd`

The original command sent to the DataHub to which this reply corresponds.

`parms`

The list of parameters sent as part of the command, as a single space-separated string, or `NULL`[1] if none are returned. If the command involved multiple parameters, the list may be truncated.

Description

This virtual method is invoked on receipt of a success message. When a client sends a command to the DataHub, the DataHub will respond with one of:

- information appropriate to the command
- a success message
- a failure message

If a command succeeds, the DataHub will either respond with information appropriate to the command, or it will send a success message if no information should be returned. If a command fails, the DataHub will always respond with a failure message.

A client has the choice of whether to receive the success messages. Commonly you don't want to expend the bandwidth by receiving success messages for every message you send to the DataHub. A client can turn on and off the success messages by emitting the DataHub command `(acksuccess 0|1)`. This command does not have an associated API function wrapper, so you have to emit it using the sendLispMessage command, like this:

```
connection.sendLispMessage (true, "(acksuccess 1)");
```

The exact syntax of the call depends on whether you are using C++, .Net or Java.

See Also

onStatusChange, onAsyncMessage, onError, onConnectionSuccess, onConnectionFailure

Notes

1. `null` in Java and C#.

III. `DataHubPoint` Methods

Table of Contents

These are the methods associated with the `DataHubPoint` class, listed alphabetically. To see the same methods grouped according to how they are used, please refer to the Categorized List of Methods.

DataHubPoint

`DataHubPoint` — constructs a DataHubPoint object.

Syntax

For C++:

```
CDataHubPoint (
   void
   );


CDataHubPoint (
   const CDataHubPoint& point
   );


CDataHubPoint (
   LPCTSTR sname
   );


CDataHubPoint (
   LPCTSTR sname,
   int itype,
   LPCTSTR svalue,
   int iconf,
   int iquality,
   int isecurity,
   int ilocked,
   int iseconds,
   int inanoseconds,
   int iflags
);
```

For Java, and C#:

```
DataHubPoint (
   String sname
   );


DataHubPoint (
   String sname,
   int itype,
   String svalue,
   int iconf,
   int iquality,
   int isecurity,
   int ilocked,
   int iseconds,
   int inanoseconds,
   int iflags
);
```

Parameters

point

An existing point to be copied.

sname

The point name. The '.' character may be used to separate name fields. A point name may be qualified by its corresponding domain as *domain*:*name*, otherwise it is treated by the DataHub as belonging to the current default domain. Point names must be matched exactly, i.e., the qualified and unqualified point names are not the same.

itype

The type of the point: PT_TYPE_STRING, PT_TYPE_INT32 or PT_TYPE_REAL.

svalue

A string representing the value, to be converted according to the specified type.

iconf

The point confidence, typically 0-100.

iquality

The point quality, typically PT_QUALITY_GOOD or PT_QUALITY_BAD.

isecurity

The point security.

ilocked

The locked flag for the point.

iseconds

The seconds component of the initial timestamp (since Jan. 1, 1970).

inanoseconds

The nanosecond component of the initial timestamp.

iflags

The point flags.

Description

Constructs a new DataHubPoint object.

See Also

DataHubPoint Class, Categorized List of Methods

~DataHubPoint

`~DataHubPoint` — destroys a DataHubPoint object.

Syntax

For C++:

```
~DataHubPoint (
   void
   );
```

For Java, and C#:

None.

Description

Destroys a `DataHubPoint` object.

See Also

`DataHubPoint` Class, Categorized List of Methods

operator=

`operator=` — assigns a new value to a DataHubPoint object (C++ only).

Syntax

For C++:

```
CDataHubPoint& operator= (
   CDataHubPoint& point
   );
```

For Java, and C#:

None.

Parameters

point

A DataHubPoint object to be copied into this DataHubPoint object.

Description

Copies the name, value and information from the source object. Existing data in the destination object is cleared. A memory exception may occur due to allocation of the name and string type data.

See Also

`DataHubPoint` Class, Categorized List of Methods, `clear`

clear

`clear` — clears the point.

Syntax

For C++, Java, and C#:

```
void clear (
   void
   );
```

Description

Clears the point, releasing the current name and string values, if any. The cleared point is a valid point of type PT_TYPE_INT32, with value of 0 and PT_QUALITY_BAD.

copy

`copy` — copies the point data.

Syntax

For C++:

```
void copy (
   const CDataHubPoint& point
   );
```

For Java, and C#:

```
void copy (
   DataHubPoint point
   );
```

Parameters

`point`

A `DataHubPoint` object to be copied into this `DataHubPoint` object.

Description

Please refer to the `operator=` method.

`getConfidence`

`getConfidence` — retrieves the user's point confidence.

Syntax

For C++, Java, and C#:

```
int getConfidence (
   void
   );
```

Returns

The confidence value of this point object.

getDateString

`getDateString` — generates a ’standard’ timestamp data/time representation.

Syntax

For C++:

```
CString getDateString (
   void
   );
```

For C#:

```
String getDateString (
   void
   );
```

Returns

A String object representing the current timestamp in exactly 26 characters as *`Day Mmm dt hr`*`:`*`mn`*`:`*`sc yyyy`*`\n`. For example `Wed Jan 02 02:03:55 1980\n`.

Description

A simple convenience function to generate a timestamp string.

getDoubleValue

`getDoubleValue` — retrieves the point data as a double-typed value.

Syntax

For C++, Java, and C#:

```
double getDoubleValue (
   void
   );
```

Returns

The current object value as a double.

Description

This method will convert the point value type if needed. If the string cannot be converted, `0.0` is returned. Strings representing hex, octal, or binary integers generate `0.0`.

getFlags

`getFlags` — retrieves the `flags` property of the point.

Syntax

For C++, Java, and C#:

```
int getFlags (
   void
   );
```

Returns

The current point flags.

`getIntValue`

`getIntValue` — retrieves the point data as an int-typed value.

Syntax

For C++, Java, and C#:

```
int getIntValue (
   void
   );
```

Returns

The current object value as an integer.

Description

This method will convert the point value type if needed. If the string cannot be converted, `0` is returned. Strings representing hex, octal, or binary integers generate `0`.

`getListeners`

`getListeners` — retrieves listeners on the point (Java only).

Syntax

For Java only:

```
LinkedList getListeners (
   void
   );
```

Returns

A list of the current listeners on the point.

getLocked

`getLocked` — retrieves the locked property of the point.

Syntax

For C++ and C#:

```
bool getLocked (
   void
   );
```

For Java:

```
boolean getLocked (
   void
   );
```

Returns

The current locked status of the point.

getName

`getName` — retrieves the point name.

Syntax

For C++:

```
CString getName (
   void
   );
```

For Java, and C#:

```
String getName (
   void
   );
```

Returns

The point name.

getNanoseconds

`getNanoseconds` — retrieves the timestamp nanoseconds component.

Syntax

For C++, Java, and C#:

```
int getNanoseconds (
   void
   );
```

Returns

The nanoseconds component of the timestamp.

getQuality

`getQuality` — retrieves the point quality.

Syntax

For C++, Java, and C#:

```
int getQuality (
   void
   );
```

Returns

The point quality.

getQualityString

`getQualityString` — generates a string representing the point quality.

Syntax

For C++:

```
CString getQualityString (
   void
   );
```

For Java, and C#:

```
String getQualityString (
   void
   );
```

Returns

A string representing the quality assigned to the point.

`getSeconds`

`getSeconds` — retrieves the timestamp seconds component.

Syntax

For C++, Java, and C#:

```
int getSeconds (
   void
   );
```

Returns

The timestamp seconds component.

getSecurity

`getSecurity` — retrieves the security property of the point.

Syntax

For C++, Java, and C#:

```
int getSecurity (
   void
   );
```

Returns

The point security.

getStringValue

`getStringValue` — retrieves the point data as a string.

Syntax

For C++:

```
CString getStringValue (
   void
   );
```

For Java, and C#:

```
String getStringValue (
   void
   );
```

Returns

A string representing the current point value.

Description

This method will convert the point value type if needed. Values of type `real` are converted using the `%g` format.

getType

`getType` — retrieves the point data type.

Syntax

For C++, Java, and C#:

```
int getType (
   void
   );
```

Returns

The point type, one of `PT_TYPE_STRING`, `PT_TYPE_REAL`, or `PT_TYPE_INT32`.

getUserdata

`getUserdata` — retrieves the user object associated with the point.

Syntax

For C++, Java, and C#:

```
void* getUserdata (
   void
   );
```

Returns

The user data previously set by `setUserData`.

qualifyName

`qualifyName` — creates a point name string qualified by a domain name.

Syntax

For C++:

```
static CString qualifyName (
   LPCTSTR domainname,
   LPCTSTR pointname
);
```

For Java and C#:

```
static String qualifyName (
   String domainname,
   String pointname
);
```

Parameters

`pointname`

The name of the point.

`domainname`

The name of the domain containing the point.

Returns

A string with the correctly formatted qualified point name.

Description

This utility method is used to add the domain name qualifier to a point name, using the format *`domainname`*`:`*`pointname`*. If the point name is already qualified, then the domain name is replaced with the new *`domainname`* specified.

If `setDefaultDomain` is used, and the point belongs to that domain, then it is not necessary to qualify the point. A qualified point name is normally used to reference points from domains other than the application's default domain.

Examples

All of these would reference the same point:

```
1. writePoint(_T("domain1:point1")), 1);
2. writePoint(CDataHubPoint::qualifyName(_T("domain1"),_T("point1")), 1);
3. setDefaultDomain(_T("domain1"));
writePoint(_T("point1"), 1);
```

See Also

setDefaultDomain, unqualifyName, writePoint, registerPoint

removeListener

`removeListener` — removes a listener from the point (Java only).

Syntax

For Java only:

```
void removeListener (
   DataHubListener listener
   );
```

Parameters

`listener`

Not yet documented.

Returns

Not yet documented.

setConfidence

`setConfidence` — sets the user's confidence in a point.

Syntax

For C++, Java, and C#:

```
void setConfidence (
   int iconf
   );
```

Parameters

`iconf`

The point confidence, typically `0`-100.

Description

Sets the user's confidence in a point. See `DataHubPoint`.

setInfo

`setInfo` — sets the information properties of a point.

Syntax

For C++:

```
void setInfo (
   CDataHubPoint& point
   );


void setInfo (
   int iconf,
   int iquality,
   int isecurity,
   int ilocked,
   int iseconds,
   int inanoseconds,
   int iflags
);
```

For Java, and C#:

```
void setInfo (
   DataHubPoint point
   );


void setInfo (
   int iconf,
   int iquality,
   int isecurity,
   int ilocked,
   int iseconds,
   int inanoseconds,
   int iflags
);
```

Parameters

point

An existing point from which to copy all the information values (not name, type or value).

iconf

The point confidence, typically `0-100`.

iquality

The point quality, typically `PT_QUALITY_GOOD` or `PT_QUALITY_BAD`.

isecurity

The point security.

ilocked

The locked flag for the point.

iseconds

The seconds component of the initial timestamp (since Jan. 1, 1970).

inanoseconds

The nanosecond component of the initial timestamp.

iflags

The point flags.

Description

Sets all the information properties of a point.

setLocked

`setLocked` — sets the locked property of the point.

Syntax

For C++ and C#:

```
void setLocked (
   bool ilocked
   );
```

For Java:

```
void setLocked (
   boolean ilocked
   );
```

Parameters

`ilocked`

The locked flag for the point.

Description

Sets the locked property of the point.

setName

`setName` — assigns a name to the point.

Syntax

For C++:

```
void setName (
   LPCTSTR sname
   );


void setName (
   LPCTSTR sdomain,
   LPCTSTR sname
);
```

For Java, and C#:

```
void setName (
   String sname
   );
```

Parameters

sname

A string containing the name to be assigned to the point.

sdomain

A string containing the domain name.

Description

Assigns a name to the point. Any existing name is released. See notes regarding *sname* parameter of `DataHubPoint`. If a domain is specified, then the qualified point name is constructed.

setQuality

`setQuality` — sets the point quality.

Syntax

For C++, Java, and C#:

```
void setQuality (
   int iquality
   );
```

Parameters

`iquality`

The point quality, typically `PT_QUALITY_GOOD` or `PT_QUALITY_BAD`.

Description

Sets the point quality.

setSecurity

`setSecurity` — sets the security property of the point.

Syntax

For C++, Java, and C#:

```
void setSecurity (
   int isecurity
   );
```

Parameters

`isecurity`

The point security.

Description

Sets the security property of the point.

setTimeStamp

`setTimeStamp` — sets the point timestamp in various ways.

Syntax

For C++, Java, and C#:

```
void setTimeStamp (
   int seconds,
   int nanoseconds
);


void setTimeStamp (
   DATE datetime
   );


void setTimeStamp (
   void
   );
```

Parameters

seconds

The number of seconds since Jan 1, 1970.

nanoseconds

The sub-second component of the timestamp, typically accurate to the extent permitted by the OS.

datetime

Time expressed in the Microsoft `DATE` format, the number of days since Dec.30, 1899.

Description

The parameter-less version of this method will set the timestamp to the current OS time.

setFlags

`setFlags` — sets the flags property of the point.

Syntax

For C++, Java, and C#:

```
void setFlags (
   int flags
   );
```

Parameters

`iflags`

The point flags.

Description

Sets the flags property of the point.

setUserdata

`setUserdata` — associates the point with a user object.

Syntax

For C++, Java, and C#:

```
void setUserdata (
   void* userdata
   );
```

Parameters

`userdata`

Any user-supplied data of size (void*).

Description

Sets the `userdata` property of a point. The user is free to set this to anything (of the appropriate size). When a point is updated in the `DataHubConnector` point cache, this property is maintained, providing the user with a facility for maintaining an association between a point and an object in the user's application space.

See Also

`getUserData`, `initializePointCache`

setValue

`setValue` — sets the point data to the specified type and value.

Syntax

For C++:

```
void setValue (
   CDataHubPoint& point
   );


void setValue (
   LPCTSTR svalue
   );


void setValue (
   double dvalue
   );


void setValue (
   int ivalue
   );
```

For Java, and C#:

```
void setValue (
   String svalue
   );


void setValue (
   double dvalue
   );


void setValue (
   int ivalue
   );
```

Parameters

point

A `DataHubPoint` class object whose data type and value are to be copied into this DataHubPoint object.

svalue

The desired string value.

dvalue

The desired double value.

ivalue

The desired integer value.

Description

Sets the object’s data value and sets the type accordingly.

setValueFromString

setValueFromString — sets the point data to the value represented by a string.

Syntax

For C++:

```
void setValueFromString (
   LPCTSTR svalue
   );


void setValueFromString (
   LPCTSTR svalue,
   int itype
);
```

For Java, and C#:

```
void setValueFromString (
   String svalue
   );


void setValueFromString (
   String svalue,
   int itype
);
```

Parameters

sValue

A string to be used to set the value.

itype

The type of the point: PT_TYPE_STRING, PT_TYPE_INT32 or PT_TYPE_REAL.

Description

If the type is specified, then the string is converted as required. If the type is not specified, then the method determines whether the string represents an integer or double value, converting to that type if possible.

unqualifyName

`unqualifyName` — removes the domain name qualifier from a point name.

Syntax

For C++:

```
static CString unqualifyName (
   LPCTSTR pointname
   );
```

For Java and C#:

```
static String unqualifyName (
   String pointname
   );
```

Parameters

`pointname`

The name of the point.

Returns

The pointname with the domain name qualifier, if any, removed.

Description

This utility method is used to remove a domain name qualifier from a point name. See `qualifyName` for more information on qualifying a point name with a domain name.

See Also

`qualifyName`, `registerPoint`

Index

Symbols

A

C

D

E

G

I

L

M

O

Q

R

S

U

W

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 written by Manuel Dias and Andrew Thomas.