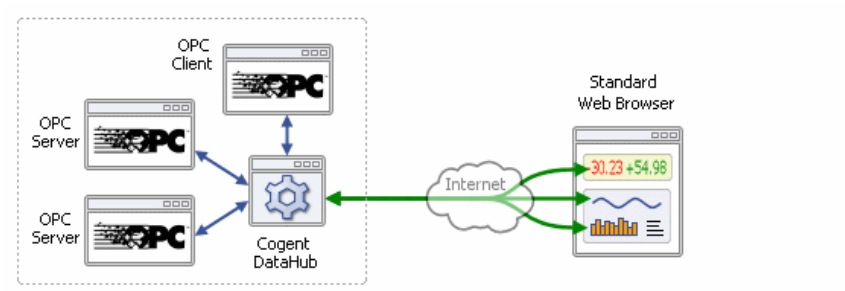
章節 7. 使用Web Server

目錄

7.1. 簡介

Cogent DataHub有內建的web server，它可以讓您根據需求使用多種不同的方式在web頁面上顯示即時資料。資料可以是來自任何連接到DataHub的資料來源。雙向資料流讓使用者檢視資料，同時也能將資料寫回DataHub。



這裡有可供使用的多種不同顯示技術:

- **ASP(Active Server Pages):** 每次client要求頁面時，DataHub會執行內嵌在HTML頁面上的DataHub指令碼。DataHub程式碼會在發送頁面給瀏覽器之前先動態建立頁面內容。這也叫做"server端指令碼"。
- **AJAX(非同步JavaScript和XML):** Javascript程式碼內嵌在一個由瀏覽器直譯的HTML頁面。資料在頁面上自動更新，而不需要重新整理頁面。這也被稱爲"client端指令碼"。
    - **Polling AJAX:** Javascript程式碼定期發送命令給DataHub以更新瀏覽器頁面的資料。
    - **Streaming AJAX:** Javascript程式碼維持一個開放的DataHub連線，這表示只要每個點的值變更，就會個別傳輸更新的值到每個點。
- **Java applets**來自**用於Java的DataHub API**，內嵌在HTML程式碼中，可製作從瀏覽器連到DataHub的個別TCP連線。一旦連線建立，小程式可以在資料值變更後立即接收和顯示更新的資料值，而無需重新整理頁面。Java applets是由瀏覽器執行，而不是由DataHub執行。

下列是這些技術的比較圖:

	ASP	Polling AJAX	Streaming AJAX	Java applets
Web瀏覽器支援	桌上型電腦和行動裝置	桌上型電腦和行動裝置	桌上型電腦和行動裝置	只支援桌上型電腦
Plug-in/Active X需求	否	否	否	是，Java plug-in
更新速度	不更新 - 需要手動更新	快速更新	極快速更新	極快速更新
系統需求 (CPU和記憶體)	極低	與Streaming AJAX和Java相較之下爲高	與Polling AJAX相較之下爲低，但和Java相同	與Polling AJAX相較之下爲低，但和Streaming AJAX相同
頻寬需求 (取決於點的數量和更新率)	極低	相對高	中等至低	中等至低
安全機制(密碼 / SSL保護)	是	是	是	是
易於使用的防火牆	是	是	是	否，需要設定防火牆
授權(除了標準的DataHub節點授權之外的需求授權)	DataHub Web Server授權	DataHub Web Server授權	DataHub Web Server授權+每個連線的TCP Link授權	DataHub Web Server授權+每個連線的TCP Link授權
程式設計需求	使用DataHub指令碼語言	使用JavaScript	使用JavaScript	使用HTML。要求Java的知識以便建立客製化的小程式。
應用程式類型常用用途	良好的顯示靜止或是緩慢變更的資料。用在輪班表和統計資料。	良好的顯示快速變更的資料和警報狀況。用在web監視和疑難排解的應用程式。	極佳的顯示快速變更的資料，用在遠端監視和診斷系統。	高速和大量使用者的最佳選擇。用於股市交易以及監視和HMI顯示的流程控制系統。

這些不同的顯示技術可以一起使用在同一個頁面上。例如，我們經常使用ASP程式碼來動態建立可以在web瀏覽器顯示即時資料的AJAX資料表。ASP程式碼爲每

個特定Data Domain裡的點，進行重複寫入資料表項目的工作。然後web瀏覽器會解譯JavaScript結果並建立相應的AJAX顯示。您還可以使用ASP**存取ODBC資料庫裡的資料**，將它顯示成爲web頁面的一部分，與DataHub的即時資料一起。

ASP的更多資訊

- 大部分的製程資料並不會快速更新，所以ASP適用於廣泛的應用程式。
- ASP通常只使用極少的系統資源和頻寬，所以對低速的連線來說是有好處的。
- ASP是處理大量使用者連線的最有效方法。
- ASP頁面是由DataHub裡面執行的指令碼建立。這代表您可以使用指令碼從其他來源，如SQL資料庫中帶入資料。接著，web頁面可以顯示兩個即時來源和資料庫裡的資料或者是文字檔案庫。
- Web頁面是由ASP指令碼建立的，它通常可以在所有桌上型電腦和移動裝置的web瀏覽器上顯示。這是因爲儘管該指令碼本身是由Gamma(即DataHub指令碼語言)撰寫的，它們所建立的web頁面仍是以plain HTML格式傳送。
- 爲了檢視ASP頁面上新的點值，您需要手動更新web頁面。

AJAX的更多資訊

- 無論何時，只要DataHub裡的值變更，AJAX就會自動地更新web頁面。
- AJAX可以處理大量使用者的高速更新。但是，server上的系統資源負荷會隨點的數量、使用者數量和更新速度而增加。
- AJAX顯示是使用JavaScript程式設計而建立的，這代表AJAX在想要全面控制web頁面上資料顯示方法的web開發人員中是很受歡迎的方法。
- AJAX可以使用最先進的桌上型電腦和移動裝置的web瀏覽器來顯示，但某些移動裝置的web瀏覽器也許不支援兩個AJAX顯示的其中一個類型。我們已經找到在移動裝置上顯示AJAX的最佳方法之一，就是Opera Mobile web瀏覽器。
- Polling AJAX和Streaming AJAX的不同之處如下:
    - Polling AJAX web頁面在一個輪詢週期中，會向DataHub要求新的資料。這代表跟Streaming AJAX相較之下，它通常有較高的CPU負載，但是在很多情況下，它的更新速度可以與之媲美。
    - Streaming AJAX web頁面在變更資料中是高效率的，這代表它與Polling AJAX應用程式相較之下，使用極少的系統資源以及少得多的網路頻寬。

Java applets的更多資訊

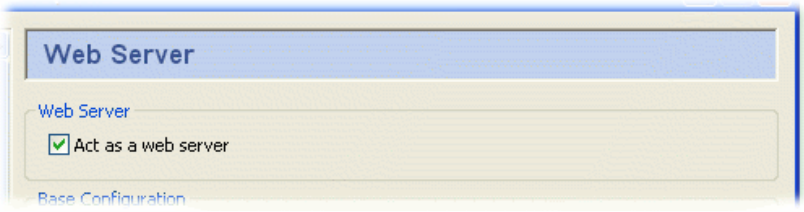
- Java applets是在複雜的web圖形，如量錶、趨勢圖和進度列中顯示高速資料的最佳選擇。
- Java顯示通常可以處理大量的資料點，並迅速更新給大量的使用者。
- 在大多數的情況下，Java web頁面會比AJAX頁面使用更少的系統及網路資源。
- Java頁面通常用於顯示管理的執行摘要螢幕，或是作爲遠端設備的HMI。
- 我們提供Java applets的廣泛範圍，以便讓您建立高階的web頁面，而無需撰寫程式碼。只要使用一個簡單的HTML編輯器來設定applets的程式碼，接著您就可以讓頁面開始運作並在幾分鐘之內執行。
- Java頁面可以使用最先進的桌上型電腦瀏覽器來顯示，但是需要先安裝免費的Java plug-in。您的系統中若尚未安裝plug-in，它會自動下載並安裝。
- 我們發現，移動裝置的web瀏覽器通常不支援Java。

7.2. 設定Web Server

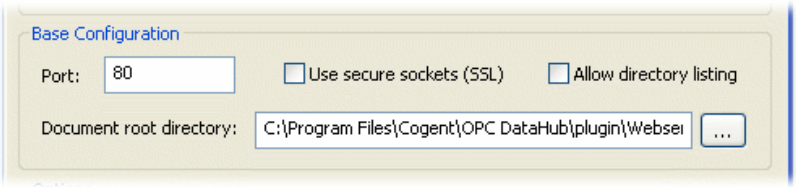
欲設定DataHub Web Server，請按照下列步驟:

1. 在DataHub執行時，DataHub系統匣圖示上點擊右鍵，接著選取**Properties**。
2. 在屬性視窗中，選取**Web Server** 

。
3. 勾選**Act as web server**方框。



4. DataHub Web Server預先設定爲執行在連接埠號碼80，但是您或許需要根據**基本配置**章節來變更此設定:

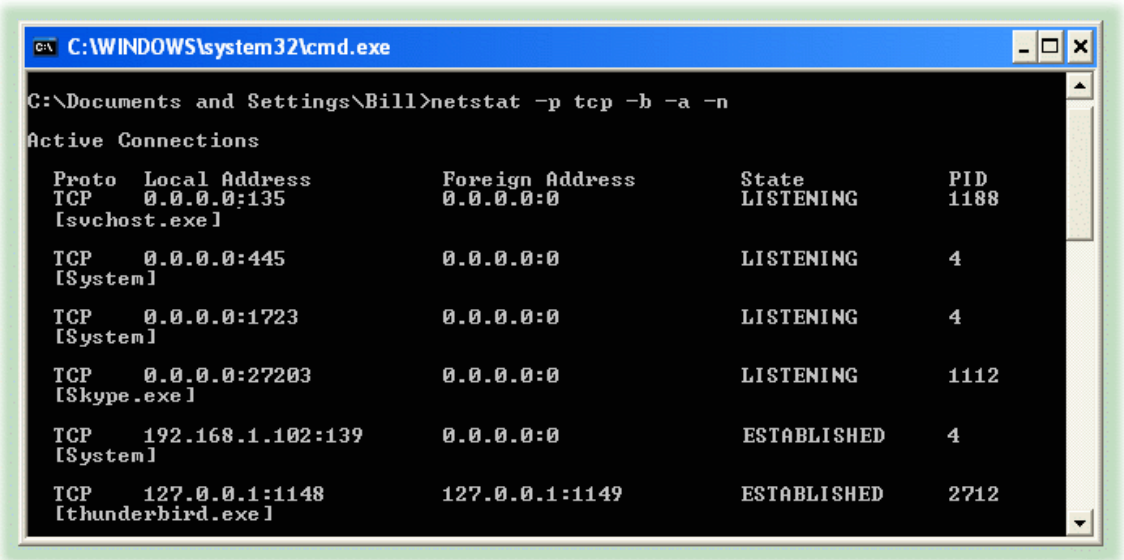


Windows允許在單一TCP連接埠裡擁有多個使用者，並且從不拒絕連線。但是，這可能會導致不規則的行爲。讓DataHub Web Server成爲連接埠的獨占使用者是極爲重要的。

欲取得您的機器裡正在使用中的連接埠清單，請按照下列步驟:

a. 在Windows **開始**功能表中，選取**執行**。
b. 輸入可執行檔的名稱**cmd.exe**並點擊**確定**。
c. 在命令提示中，輸入:

```
netstat -p tcp -b -a -n
```



該結果會顯示TCP通訊協定的資料表和所有正在使用中的可執行檔名稱。以下有兩個值得關注的欄位: Local Address和State。在Local Address中，最後面的號碼(冒號後面)是該process正在使用的連接埠號碼。State欄位顯示該process的狀態。我們唯一關注的狀態就是LISTENING。無論你對DataHub Web Server使用哪個連接埠，它都應該是該連接埠唯一的process。

如果您有一個或更多程式正在聽候或者是建立在與Cogent DataHub相同的連接埠上，您有兩個選擇:

- 變更DataHub Web Server的連接埠號碼(如上所示)，或是
- 變更其他每個正在使用該連接埠的其他程式的連接埠號碼。

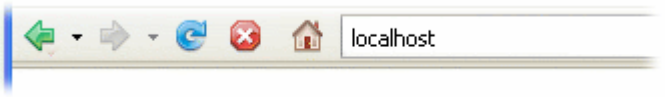
連接埠號碼1至1024是保留的，而連接埠80，舉例來說，是爲HTTP保留的，這就是爲什麼我們把它作爲DataHub Web Server的預設連接埠。如果您欲變更DataHub Web Server的連接埠號碼80，我們建議將其設定爲1025至65535號碼之間。

7.3. 檢視Web示範

Click here to watch a video.

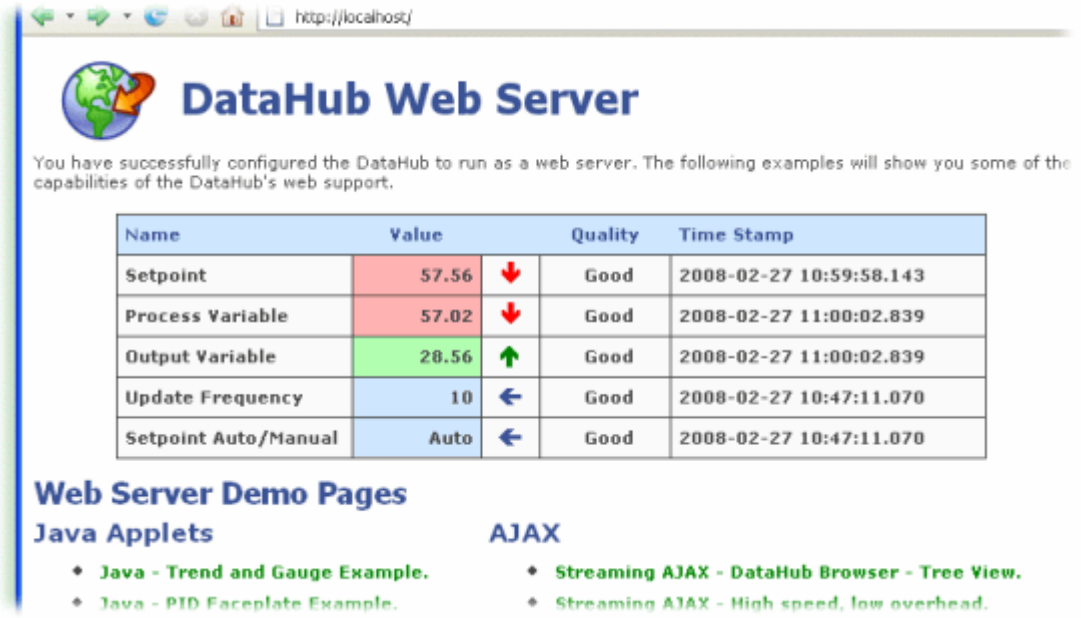
以下教您如何存取隨附在DataHub安裝的示範頁面:

1. 確保DataHub正在執行，並且已經啓動DataHub Web Server。
2. 將DataHub的IP位址輸入您的web瀏覽器上方的Address欄位。



如果您將DataHub和web瀏覽器執行在同一台機器上，輸入localhost。否則，就輸入執行DataHub電腦的IP位址或是名稱。

3. 會出現歡迎畫面。您可以點擊連結以檢視各種演示。演示中的即時資料來自DataPid，如果沒有看到資料更新，您就必須啓動DataPid。



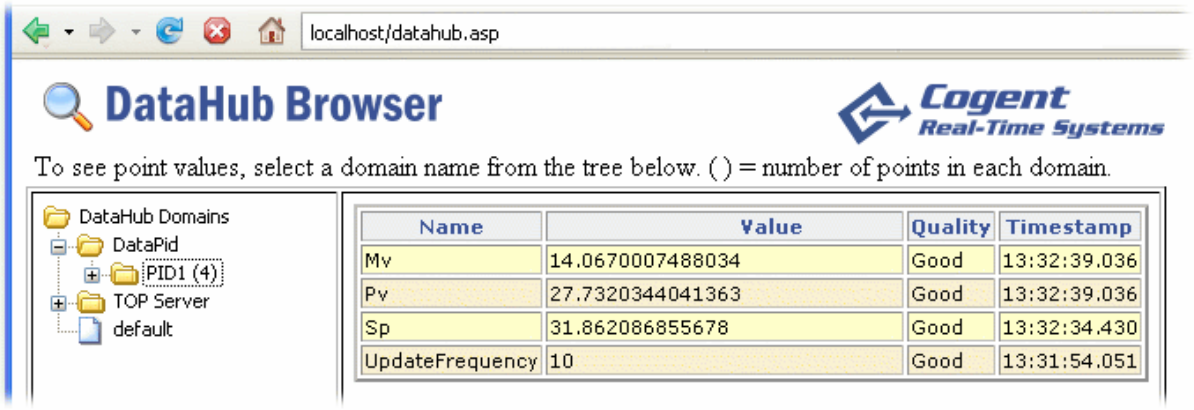
7.4. 檢視您自己的資料

7.4.1. DataHub Browser

Click here to watch a video.

您可以使用DataHub Browser檢視目前DataHub裡的所有資料點。以下是檢視您系統中資料的方法:

1. 確保DataHub正在執行並且連接到資料來源，以及已經啓用DataHub Web Server。
2. 如果您將web瀏覽器執行在與DataHub相同的機器上，在您瀏覽器上方的Address欄位輸入localhost/datahub.asp。如果您將web瀏覽器執行在與DataHub不同的機器上，請輸入執行DataHub電腦的IP位址或是電腦名稱，而不是輸入localhost。
   Web Data Browser頁面應該會顯示如下:



   DataHub domains和資料階層顯示在左窗格中，指定部分階層裡的點值即時更新會在右側窗格顯示。您可以在檔案C:\Program Files\Cogent\Cogent DataHub\Plugin\Webserver\html\domaintreeheader.asp中，修改此頁面的標題。

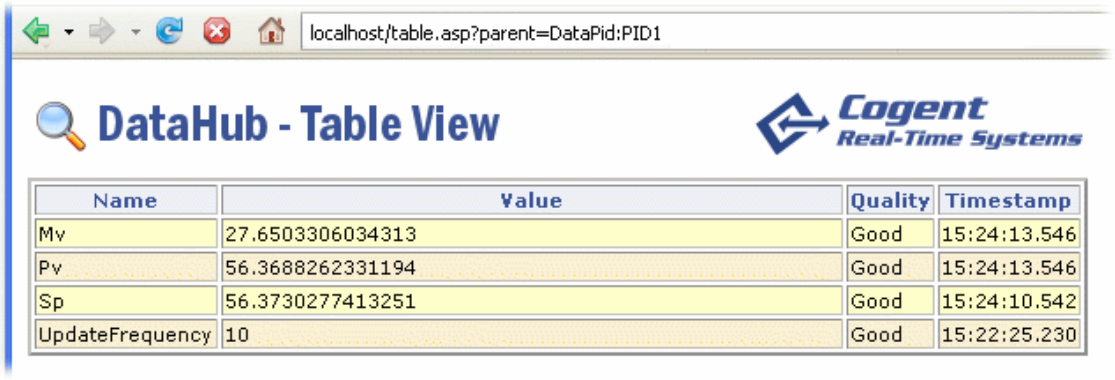
7.4.2. DataHub Table View

您也可以使用DataHub Table View頁面在您系統裡的單一資料網域檢視所有的資料點，方法如下:

1. 確保DataHub正在執行並且已連接到您的資料來源，還有確保已啓動DataHub Web Server。
2. 開啓此電腦或是其他client電腦的web瀏覽器，並且在網址列中輸入下列URL:

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

   where
   computer address
   執行Cogent DataHub之電腦的IP位址或網路名稱。
   domain name
   包含您欲顯示之資料的Data Domain名稱。
   branch
   分支(如果有的話)包含您欲顯示的資料。
   sub-branch
   子分支(如果有的話)包含您欲顯示的資料。
   有時候Cogent DataHub裡的資料組織可以是很複雜的，包含分支和子分支。這種情況最常出現在從一個擁有多層級資料階層架構的資料來源讀取資料時。在這種情況下，我們使用分支和子分支語法(如上所示)來參照資料結構裡的位址。
3. 範例:
   - 如果您正在執行DataPid程式，會顯示此URL: `http://localhost/table.asp?parent=DataPid:PID1`如下:



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

- 輸入項目如下:

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

  將會顯示來自DataHub的Plant_1資料網域中的資料。

- 輸入項目如下:

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

  將會顯示DataHub的Section_1資料網域中Zone_1分支的資料。

- 輸入項目如下:

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

  將會顯示Setpoints分支裡的資料，位於DataHub的Section_1 Data Domain中數狀結構內下三層。它位於DataHub的Section_1 Data Domain中樹狀結構內下三層。

- 至於其他的輸入項目，您可以只輸入**localhost/...**，如果您正在DataHub電腦中執行web瀏覽器，則輸入如下:

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

4. 您可以藉由編輯檔案C:\Program Files\Cogent\Cogent DataHub\Plugin\Webserver\html\table.asp以變更標題圖像。變更的行列會明確地標示:

   ```
   <!-- *************** Logo Image Goes Here *************** -->
   <img src="images/DataHubTable.gif" width="600" height="40" vspace="10" border="0" /><br />
   <!-- ************************************************** -->
   ```

5. 您也可以在資料表下方新增頁面內容，藉由在接近頁面下方，最後的</script> tag的下方放入程式碼。

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

   (insert extra HTML here)

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

7.5. 修改ASP和AJAX示範檔案


欲修改Java演示，您需要使用**用於Jave的DataHub API**。

您可以使用Web Server檔案庫裡的演示檔案作爲建立您自己的web頁面的基準。預設的位置如下:

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

ASP和AJAX都使用.asp檔案來建立web頁面。在DataHub distribution裡的.asp檔案包含內嵌的DataHub指令碼，此指令碼撰寫部分轉譯的.asp最後頁面。換句話說，在當您在瀏覽器中選取**View Source**所看到的並不是原始的.asp檔案，只是處理後的結果。舉例來說，DH_asp_2.asp檔案裡的這行程式碼:

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

撰寫如上，但在DH_asp_2.asp頁面顯示如下:

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

每個.asp檔案可以包含兩個特殊標記語法的類型以便插入DataHub指定碼的程式碼，每個都有其特殊用途。

- **<@ ... @>** 指出建立頁面時被評估的表達式。表達式的結果被寫入此頁面。這裡的例子會被插入DataHub點的值裡。
- **<% ... %>** 指出建立頁面時也被評估的一個表達式或陳述式，*但是*完全沒有寫入此頁面。例子可以是在頁面中被其他程式碼使用的函數。

範例

DH_asp_2.asp檔案顯示如下。它包含兩個函式: TimeStamp，以計算時間戳記；以及NewSetPoint以便將Setpoint新的值發送回DataHub。它也包含一個for的loop以便建立表格列。所有使用<% ... %>特別標示的語法，當值在表格中顯示時，會使用<@ ... @>語法特別標示。

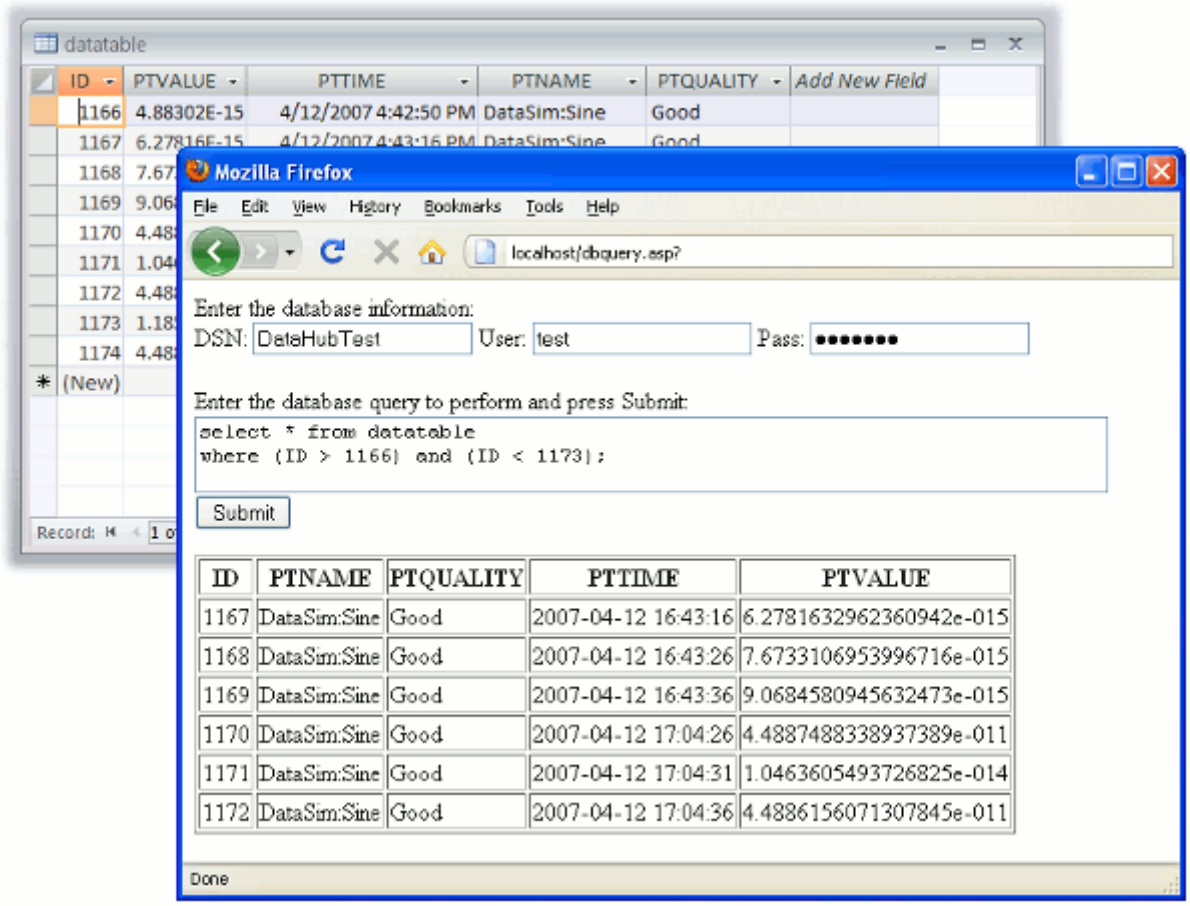

該指令碼也使用陳述式的try / catch以便進行錯誤處理。欲了解更多資訊，請參閱Gamma文件集的**try catch**章節。

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

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

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

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

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

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

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

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

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


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

7.6. 使用ASP來查詢資料庫和顯示結果

使用Cogent DataHub的**ODBC scripting**功能，它可以從web頁面發送一個SQL查詢到資料庫，並且在頁面上檢視結果。dbquery.asp演示頁面包含在DataHub distribution裡，它提供一個範例以告知它是如何完成的。藉由在DataHub執行您的本機機器，您可以藉由在web瀏覽器輸入localhost/dbquery.asp來檢視此頁面:



您只需爲此資料庫輸入一個**DSN**、使用者名稱和密碼(如果有的話)，以及一個SQL查詢。當您按下**Submit**按鈕時，頁面會發送一個查詢給DataHub，它會傳遞查詢給資料庫，並返回結果，在此頁面上的表格中顯示。

此頁面的來源檔案與其他所有的ASP檔案一起包含在您的DataHub distribution裡:

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

以下是頁面來源的副本:

```
<html>
<head>
<%
/*
 * This file presents an entry form for the user to specify a
 * DSN, user name, password and SQL query to be executed by the
 * DataHub.  The result is displayed as a table in the user's
 * browser.
 *
 * Normally, the DSN, user name and password would be hard-coded
 * into the Gamma portion of this ASP file.  The ASP file is
 * stored on the server running the DataHub, and all Gamma code
 * is stripped from the file and executed before the result is
 * returned to the user.  Therefore the DSN, user name and
 * password are never transmitted across the network to the user.
 */

require ("AJAXSupport");
```

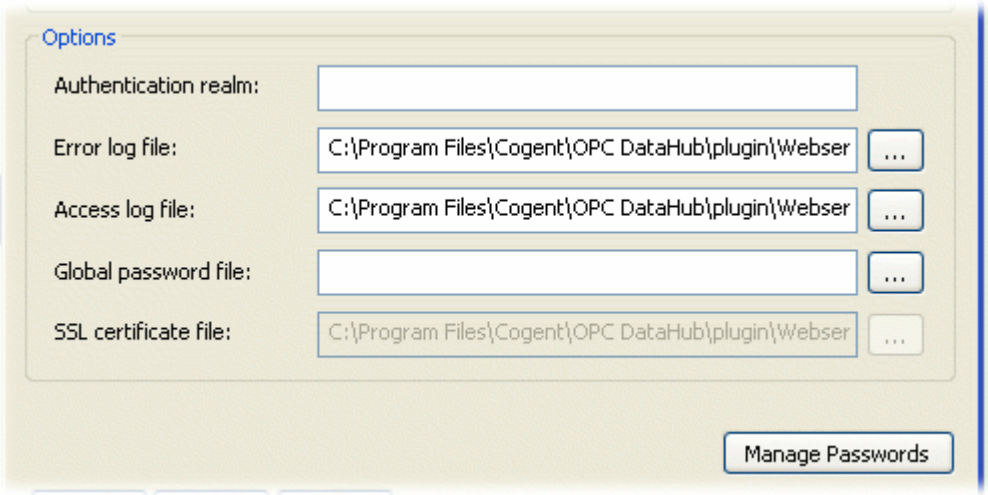
```
require ("ODBCSupport");

local cmpfn, getArg, sortfn;

/* Set up some function for quickly accessing the URL arguments */
function cmpfn(x,y) { strcmp(x[0],y); }
function sortfn(x,y) { strcmp(x[0],y[0]); }
function getArg(name, dflt)
{
    local    arg = bsearch(_vars, name, cmpfn);
    if (arg && !undefined_p(car(arg)))
        arg = car(arg)[1];
    else
        arg = dflt;
    arg;
}
_vars = sort(_vars,sortfn);

/* Read the input URL arguments */
local DSN = getArg("DSN","Enter DSN");
local Password = getArg("Password","Enter Password");
local Username = getArg("Username","Enter Username");
local Query = getArg("Query","");

/* Connect to the database */
local env, conn, Connect, DoQuery;

function Connect ()
{
    local   ret;

    /* Create the ODBC environment and connection */
    env = ODBC_AllocEnvironment();
    conn = env.AllocConnection();

    /* Attempt the connection. */
    ret = conn.Connect (DSN, Username, Password);
    if (ret != SQL_SUCCESS && ret != SQL_SUCCESS_WITH_INFO)
        error (conn.GetDiagRec());
}

function DoQuery()
{
    local    result = conn.QueryToTempClass(nil, Query);
}

%>
</head>

<body>

Enter the database information:
<form>
DSN: <input type="text" id="DSN" name="DSN" value="<%= DSN %>">
User: <input type="text" id="Username" name="Username" value="<%= Username %>">
Pass: <input type="password" id="Password" name="Password" value="<%= Password %>">
<br>
<br>
Enter the database query to perform and press Submit:
<br>
<textarea name="Query" wrap="logical" rows="10" cols="80">
<%= Query %></textarea>
<br>
<input type="submit" id="Submit" value="Submit">
</form>

<%
try {
if (Query != "")
{
    Connect();
```

```
    local result = DoQuery();
    if (!result || length(result) == 0)
    {
    }
    else
    {
        local     first = result[0];
        %> <div style="overflow: auto; width: 660; height: 300"> <%
        %> <table border="1"> <%
        %> <tr> <%
        with var in instance_vars(first) do
        {
            %> <th><%= car(var) %></th> <%
        }
        %> </tr> <%
        with row in result do
        {
            %> <tr> <%
            with var in instance_vars(row) do
            {
                %> <td><%= cdr(var) %></td> <%
            }
            %> </tr> <%
        }
        %> </table> <%
        %> </div> <%
    }
}
} catch {
%>
    A problem occurred while running server-side scripts on this page:<br>
    <%= _last_error_ %>
<%
}

/* Do a little cleanup.  The garbage collector would eventually get to
   this, but we can be kind and unwind. */
if (conn)
{
    conn.Disconnect();
    destroy (conn);
}
if (env)
    destroy (env);
%>

</body>
</html>
```

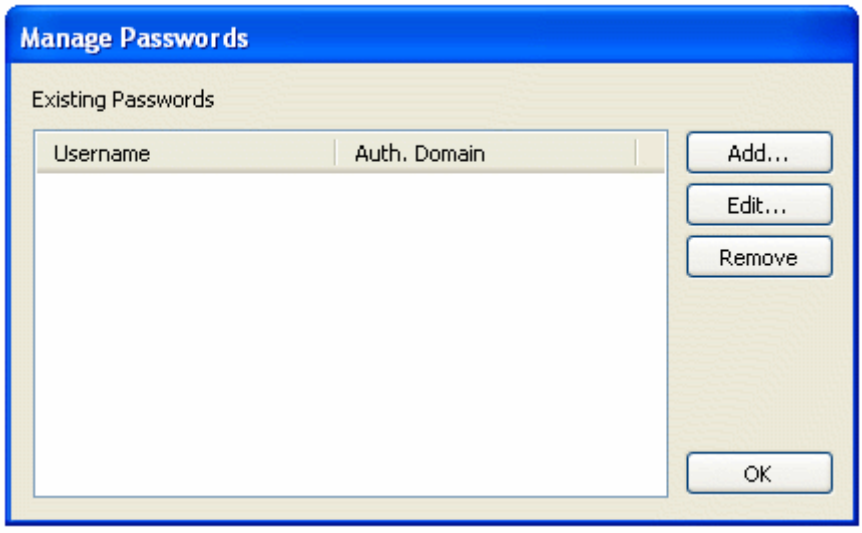
7.7. 建立密碼

您可以爲該系統建立全域的密碼，以便控制整個網站的存取。方法如下:

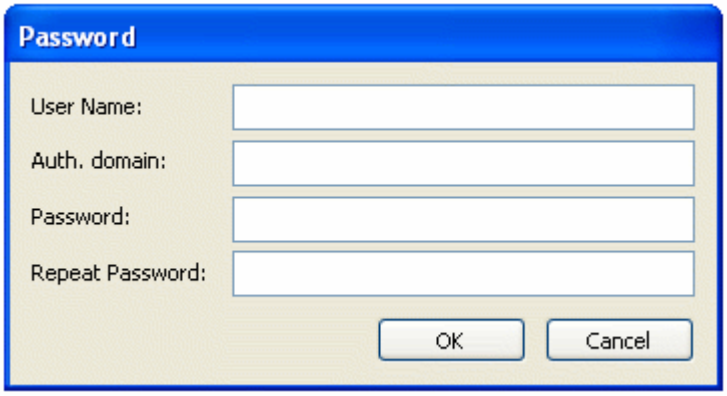
1. 在**Options**區域的**Authentication realm**欄位中輸入一個名稱。這將會顯示在密碼提示的對話框中。



2. 在**Global password file**輸入一個檔案名稱。在第一次，您將需要在您所選取的資料夾裡建立一個空白檔案。我們建議使用C:\Program Files\Cogent\Cogent DataHub\require\資料夾。若在這裡輸入一個檔案名稱而且檔案存在時，DataHub會要求每個使用者輸入他們的使用者名稱和密碼(如同下列設定)才能存取該網站。欲停止這種行爲，您必須在這裡移除輸入的檔案名稱，或者是將其更改爲一個不存在的檔案。
3. 點擊**Manage Passwords**按鈕以開啓**Manage Passwords**視窗:



4. 點擊**Add**或**Edit**按鈕以開啓**Password**視窗:



5. 輸入下列資訊:

User name
輸入使用者名稱，不包含空白鍵。

Auth. realm
輸入您在**Options**項目(見上文)的**Authentication realm**欄位中輸入的名稱。

Password
輸入新的密碼，不包含空白鍵。

Repeat Password
再次輸入密碼，必須完全跟密碼相同，接著點擊**OK**。

6. 點擊屬性視窗下方的**Apply**按鈕以套用變更。

7.8. 從Cogent DataHub建立XML輸出

Cogent DataHub用幾種方式製作XML格式的資料給web式的client。機制的選取將取決於client應用程式的需求。一般來說，這些機制可以被分類成串流或輪詢方法兩者之一。

串流XML資料

Cogent DataHub Web Server提供一種有效的方法讓串流資料通過TCP/IP的通訊端。最初的通訊端連線使用HTTP/HTTPS交涉。一旦通訊端建立完成，Cogent DataHub裡的一個獨立執行緒會接管此通訊端的責任。在最初的HTTP交涉後，所有的通訊都是單向的。也就是說，DataHub發出資料給client應用程式，但是client不會傳輸任何資料。XML格式可以被預先定義或使用者可設定的。實際的資料格式化是透過DataHub裡執行的指令碼語言，Gamma所執行的。

串流XML機制使得某些在Cogent DataHub提供"串流AJAX"的基礎科技變得可用。只有資料格式是不同的。

輪詢XML資料

Cogent DataHub Web Server提供兩種方法可以透過輪詢接收XML資料。在一般情況下，Cogent DataHub提供一個特殊的URL，它可直接存取DataHub引擎裡的資料集以便建構一個XML字串。XML格式是預先定義的。字串建構完全在記憶體內執行，所以這種方法非常有效。

如果client要求一個不同的XML格式，這可以經由一個ASP頁面提供。DataHub使用Gamma作爲其ASP語言，也就是說請求一個web頁面可能會觸發Gamma指令碼呼叫，這會產生一個web開發人員需求的任何格式的XML資料。

7.8.1. Streaming XML基本指南

7.8.1.1. 內建串流資料

內建的串流資料使用這種表單的URL: http://*hostname:port/stream?arguments*來進行存取。引數是由&符號分隔，而且可以任意組合:

`name=`*`pointname1|pointname2|...`*

這是個別點名稱的清單，可以擷取自DataHub。點的名稱完全完整，如DataSim:Sine，而不是只有Sine。點的名稱之間使用直立線符號"|"分隔。如果名稱的引數被省略，那麼就必須提供domain的引數。例如:

```
name=DataSim:Sine|DataSim:Ramp|DataSim:Square
```

`template=[json,jsonp,djson,djsonp,xml,lisp]`

這會從清單裡顯示的可用選項中選取輸出格式。Template會決定預設的head、tail、prefix和suffix。例如:

```
template=xml
```

`head=`*`file name`*

其標題是相對於DataHub Web Server文件根目錄的檔案名稱，一旦首次的連線建立，它就會被傳輸。這可能包含像是XML版本的資訊或是web首頁的資訊，例如:

```
head=my_header_file
```

`tail=`*`file name`*
tail是相對於DataHub Web Server文件根目錄的檔案名稱，它會在連線終止之前立即傳輸。當達到msglimit時，DataHub會自動終止連線。如果msglimit是0，此檔案絕對不會被傳輸。

`prefix=<`*`any string`*`>`

prefix XML tag會在每個資料更新之前傳輸。DataHub會在它可供使用時傳輸資料，以節流處理設定而定。會爲資料點的每個群組傳輸一次前置字元，而不是每個資料點傳輸一次。例如:

```
prefix=<points>
```

`suffix=</`*`any string`*`>`

XML tag suffix會在每個資料更新之後傳輸。請參閱上面的prefix以了解更多細節。例如:

```
suffix=</points>
```

`msglimit=`*`a non-negative integer`*

當連線開啓時，爲了清除累積的資源，某些clients需要定期關閉並重新開啓一個串流連線。這會讓client指示在DataHub必須關閉連線之前有多少資料更新可以被傳輸。一旦連線關閉，client要負責重新開啓連線。如果值是0，Cogent DataHub不會刻意關閉連線。DataHub只會在連線關閉前傳輸tail檔案，所以如果此值爲0，tail檔案絕對不會被傳輸。例如:

```
msglimit=10000
```

`throttle=`*`a non-negative floating-point number`*

在發送資料之前等候的秒數，讓client請求Cogent DataHub的最大更新率。也就是說，如果資料變更比throttle秒數還要快，DataHub會累積資料點，並且只在上次傳輸後的throttle秒數通過之後才會進行傳輸資料點。如果一個資料點在這個時期內變更不只一次，就只會傳輸最新的值。把throttle設定爲0以指示Cogent DataHub傳輸所有的資料而沒有延遲。例如:

```
throttle=0.25
```

`user=`*`a non-empty string`*

如果Cogent DataHub安全設定需要一個TCP連線的使用者名稱和密碼，client可以在此支援。例如:

```
user=my_name&pass=my_pass
```

`pass=a non-empty string`
見上文。
`domain=a domain name list`

client可能會要求Data Domain裡的所有資料，而不是個別的已命名資料點。如果提供domain引數，就不會提供name引數。多個domain名稱可以由直立線符號"|"分隔。例如:

```
domain=DataPid
```

> 因爲Cogent DataHub的bug，這種引數目前無作用，改爲使用: `name=domain_name&children=1&recursive=1`。 例如: name=DataPid&children=1&recursive=1

`children=[0,1]`

當client提供一個名稱引數，它也可以支援children引數以便指示是否也讀取出此點的任何子資料點。如果名稱引數支援多個點，children引數就會套用到所有的點。0的值表示不讀取children。1的值表示讀取children。它也會受到下列recursive設定進一步的影響。例如:

```
children=1
```

`recursive=[0,1]`

如果children引數是1，那麼recursive的值會決定是否在命名點上開始搜尋資料階層，擷取所有命名點的子代。如果recursive的值是0，就只有每個點的下層子系會被擷取。如果recursive的值是0，那麼命名點的所有子代就會被擷取。例如:

```
recursive=1
```

範例

欲取出DataPid Domain的所有點，用最大更新率5Hz，以最大的10000資料變更，URL會是:

```
http://localhost/stream?name=DataPid&children=1&recursive=1&msglimit=30&throttle=0.2&template=xml
```

輸出結果如下:

```
<points>
<point name="DataPid:PID1.Controller.Kd" value="0.01" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Controller.Ki" value="0.5" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Controller.Kp" value="0.25" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Mv" value="37.2139761632173" type="1" quality="192"
timestamp="1275683572.9200001" />
<point name="DataPid:PID1.Plant.Ki" value="0.5" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Plant.Kp" value="2" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Pv" value="55.551238388592" type="1" quality="192"
timestamp="1275683572.9200001" />
<point name="DataPid:PID1.Range.Amplitude" value="100" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Range.Offset" value="50" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Setpoint.AutoMode" value="1" type="2" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Setpoint.AutoTime" value="5" type="1" quality="192"
timestamp="1275682823.9979999" />
<point name="DataPid:PID1.Setpoint.SpInput" value="0" type="1" quality="192"
timestamp="-2147483648" />
<point name="DataPid:PID1.Sp" value="72.4715414899136" type="1" quality="192"
timestamp="1275683572.1540003" />
<point name="DataPid:PID1.UpdateFrequency" value="10" type="1" quality="192"
timestamp="1275682823.9979999" />
</points>
<points>
<point name="DataPid:PID1.Mv" value="38.3540008662675" type="1" quality="192"
timestamp="1275683573.1379995" />
<point name="DataPid:PID1.Pv" value="57.5623437038486" type="1" quality="192"
timestamp="1275683573.1379995" />
</points>
<points>
<point name="DataPid:PID1.Mv" value="39.2907100165968" type="1" quality="192"
timestamp="1275683573.3569999" />
<point name="DataPid:PID1.Pv" value="59.5695627360927" type="1" quality="192"
timestamp="1275683573.3569999" />
</points>
<points>
<point name="DataPid:PID1.Mv" value="40.0353526249595" type="1" quality="192"
timestamp="1275683573.5760002" />
<point name="DataPid:PID1.Pv" value="61.5352593843648" type="1" quality="192"
timestamp="1275683573.5760002" />
</points>
<points>
<point name="DataPid:PID1.Mv" value="40.6012237044704" type="1" quality="192"
timestamp="1275683573.7950006" />
<point name="DataPid:PID1.Pv" value="63.4279691691699" type="1" quality="192"
timestamp="1275683573.7950006" />
</points>
```

7.8.1.2. 客製化內建的串流資料

控制串流web資料格式的是DataHub安裝目錄裡require\WebstreamSupport.g提供的Gamma指令碼。您可以新增您自己的格式範本，以便讓您指定資料在您的程式裡顯示的方式。例如，預設的XML版面配置如下:

```
<points>
<point name="DataSim:Sine" value="0.5" type="1" quality="192" timestamp="1275683573.7950006" />
</points>
```

沒有前端或尾端的字串。前置字元字串是<points>，後置字元字串是</points>。除此之外，還有三個可用的格式字串可以決定如何將不同的資料類型格式化。

`stringformat` - 一個將被用在準備字串資料的格式字串。
`numberformat` - 一個將被用在準備數值(實數和整數)資料的格式字串。
`arrayformat` - 一個將被用在準備陣列資料的格式字串。

在預設XML範本，這些是:

`stringformat` - `<point name=%s value=%w type="%d" quality="%d" timestamp="%g" />`
`numberformat` - `<point name=%s value=%s type="%d" quality="%d" timestamp="%g" />`
`arrayformat` - `<point name=%s value=%s type="%d" quality="%d" timestamp="%g" />`
`separator` - 一個將被用在分隔單一組前置字元和後置字元字串之間的字串。例如","。

每個格式的字串必須爲5個輸入參數編碼。依次爲:

`name` - 顯示完整點名稱的字串。
`value` - 點的值，它可能會是字串、號碼或陣列之一。
`type` - 顯示資料類型的號碼，0=字串或陣列，1=浮點數量，2=整數量。
`quality` - 以整數表示的OPC連線品質。
`timestamp` - 一個UNIX時間戳記作爲秒數，始於1970年1月1日UTC。

陣列已編碼作爲表單"['value1','value2','value3',...]"的字串。

藉由修改Gamma class WebstreamSupport的定義來指定這些參數。最初的文件可以在此檔案中找到: require\WebstreamSupport.g。最好不要修改此檔案，而是在一個單獨的檔案裡新增您的修改，如下所示:

1. 建立您自己Gamma檔案，例如MyCustomWebstream.g。
2. 移除此檔案裡現存的所有程式碼。
3. 插入下列程式碼:

```
require ("WebstreamSupport");

// Ensure that there is an instance variable for my new template
if (!has_ivar(WebstreamSupport, #my_custom))
{
    class_add_ivar(WebstreamSupport, #my_custom);

    // Re-instantiate the helper instance with the new instance variable included
    Webstream = ApplicationSingleton (WebstreamSupport);
}
Webstream.my_custom = new WebstreamTemplate ("CUSTOM", "", "my_custom",
                                             "<data>\n",
                                             "</data>\n",
                                             0, 0.0,
                                             "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n",
                                             "",
                                             "<point name=%s value=%w type=\"%d\" quality=\ %d\" timestamp=\"%g\" />\n",
                                             "<point name=%s value=%s type=\"%d\" quality=\"%d\" timestamp=\"%g\" />\n",
                                             //"<point name=%s value=%s type=\"%d\" quality=\"%d\" timestamp=\"%g\" />\r
                                             " ");
```

4. 修改程式碼以符合您的需求。
   程式碼在class WebstreamSupport裡建立一個新的範本成員，接著建立一個WebstreamTemplate以便指定一個參數組給該範本。WebstreamTemplate的建構函式採用下列引數:
   `tplname` - 任意的字串，不使用。
   `pointnames` - 一個預設直立線符號點名稱的字串，應爲""。
   `template_id` - /stream URL中提供的範本名稱。
   `prefix` - 每個資料變更群組的前置字元字串。
   `suffix` - 每個資料變更群組的後置字元字串。
   `msglimit` - 預設的msglimit如上。
   `throttle` - 預設的throttle如上。
   `head` - 標題字串(非以上的檔案名稱)，一旦連線後會被傳輸。
   `tail` - 在DataHub中斷連線之前立即傳輸一個尾端字串。
   `stringformat` - 如上。
   `numberformat` - 如上。
   `arrayformat` - 如上。根據您的OPC DataHub而定，該引數可能會遺失。
   `separator` - 如上。
5. 確保DataHub啓動時，有執行您的新指令碼。
6. 執行您的指令碼。您現在應該可以從URL(如下)查詢資料:

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

7.8.2. 輪詢XML基本指南

典型的AJAX應用程式透過輪詢來擷取資料，其使用一個名爲XmlHttpRequest的Javascript函式。此函式製作一個非同步HTTP請求到一個URL，這會傳回一個XML文件。Javascript應用程式會分析這個XML文件以摘錄相關資訊。

7.8.2.1. 內建的HTTP資料輪詢

Cogent DataHub實做一個內建的XML輪詢機制，它建構一個完全存在記憶體裡的XML文件以減少CPU使用率。此文件的格式不是使用者可設定的，而且當指定template=xml時，格式會與串流資料設備產生的輸出類似。

內建的串流資料使用此表單進行存取: http://hostname:port/points?arguments。引號是由&符號分隔的，且可任意組合:

`name=pointname1|pointname2|...`

這是個別點名稱的清單，可以擷取自DataHub。點的名稱完全完整，如DataSim:Sine，而不是只有Sine。點的名稱之間使用直立線符號"|"分隔。如果名稱的引數被省略，那麼就必須提供網域的引數。例如:

```
name=DataSim:Sine|DataSim:Ramp|DataSim:Square
```

`domain=a domain name list`

client可能會要求資料網域裡的所有資料，而不是個別的已命名資料點。如果提供domain引數，就不會提供name引數。多個網域名稱可以由直立線符號"|"分隔。例如:

```
domain=DataPid
```

> 因爲Cogent DataHub的bug，這種引數目前無作用，改爲使用: `name=domain_name&children=1&recursive=1`。例如: name=DataPid&children=1&recursive=1

`style=[json,jsonp,djson,djsonp,xml,lisp]`

這會從清單裡顯示的可用選項中選取輸出格式。例如:

```
style=xml
```

`prefix=<any string>`

prefix XML tag會在每個資料更新之前傳輸。DataHub會在它可供使用時傳輸資料，以節流處理設定而定。會爲資料點的每個群組傳輸一次前置字元，而不是每個資料點傳輸一次。例如:

```
prefix=<points>
```

`suffix=</any string>`

XML tag suffix會在每個資料更新之後傳輸。請參閱上面的prefix以了解更多細節。例如:

```
suffix=</points>
```

7.8.2.2. 客置化HTTP資料輪詢

因爲內建的資料輪詢不是使用者可設定的，所以它不是建立客置化資料格式的適合機制。取而代之的，客置化XML資料可以使用由使用者所提供的ASP頁面發出。Cogent DataHub Web Server內的ASP機制使用Gamma指令碼語言作爲其ASP語言。這表示一個ASP頁面擁有存取權，可以存取DataHub裡的即時資料和任何Gamma函式如資料庫calls。Gamma calls是用來建立插入頁面的字串。

可建立一個XML的一簡單ASP頁面看起來如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<points>
    <point name="DataSim:Sine" value="<%= $DataSim:Sine %>" />
</points>
```

在這個範例中，文件的架構是簡單的XML，內容是<%= %>分隔符號內嵌的Gamma calls裡的資料。評估Gamma表達式的結果是轉換成一個字串，接著插入檔案裡，取代<%= %>分隔符號和分隔符號之間所有的文字。這些分隔符號之間的Gamma指令碼必須是一個單一Gamma表達式，而非陳述式。例如: 表達是2+2是合法的，但陳述式a=2+2是不合法的。

一個更複雜的範例可以在Gamma裡實作loop以查詢DataHub裡的資料集並爲每個資料點提供一個值。

```
<?xml version="1.0" encoding="UTF-8"?>
<points>
<%
local points = datahub_points("DataPid", nil);
with point in points do
{
    if ((point.flags & 0x30) == 0) // ensure point is not an assembly
    {
        %>
        <point name="<%= point.name %>" value="<%= point.value %>" />
        <%
    }
}
%>
</points>
```

在這個範例中，分隔符號<% %>是用來指出不會建立一個替換字串的Gamma指令程式碼。<% and %>之間的Gamma指令碼必須有一個或更多Gamma陳述式，而非表達式。注意Gamma陳述式可能會打破<% %>分隔符號的橫跨發生次數，允許一個自然機制以便指定檔案裡的XML部分。

ASP檔案可以被放進任何DataHub Web Server主目錄裡面的任何子目錄。client應用程式接著可以簡單地在此檔案裡重複讀取，以擷取目前的資料值。