How to Configure FSGateway and RSLinx Data Source

LEGACY TECH NOTE # 392

Doc ID: TN64 **Version**: 6 **Status**: published **Last Modified**: 05/01/2015 **Cat**: FS Gateway

SUMMARY

FSGateway can be configured to access data sources using DDE, SuiteLink, and OPC protocols.

This technote explains the step-by-step procedure to configure the FSGateway and the Rockwell Software's RSLinx data source to access data in an Allen-Bradley PLC (a ControlLogix PLC is being used in this example).

Before you continue, make sure you have the following:

- Install and configure RSLinx (OEM version minimum, Professional or higher recommended) so that it communicates with the PLC. RSLinx needs to have at lease one topic defined.
- Check the Readme file for the system requirements and the installation information before installing the FSGateway.
- Install the latest version of the FSGateway. If a previous version was installed, be sure to uninstall it using Control Panel - Add/Remove Programs. To check the version number of the server, use Control Panel - Add/Remove Programs - select Wonderware ABCIP DAServer, then click **Support Information**.

SITUATION

Assumptions

- FSGateway version **1.0.001** is used in this technote.
- RSLinx Gateway version 2.42.00 (Build 18) is used.
- Both FSGateway and RSLinx are installed on the same computer with Windows®2000 Professional Service Pack 4.
- This technote assumes the user has a basic working knowledge and understanding of Allen-Bradley software/hardware, Microsoft® Operating System, and Wonderware® products. If you have problem configuring the ControlLogix or RSLinx, please contact Allen-Bradley for assistance at www.ab.com
- The configuration of the ControlLogix PLC project is outside the scope of this tech note. However, typical Control Scope tags have been added in the PLC: **MyInt**, **MyFloat**, and **MyBool**. These tags are of types integer, floating point (or real), and Boolean respectively.

Configure RSLinx Data Source

1. Launch RSLinx.
2. Verify that the communication driver has been added, correctly configured, and is running.

   In this example, the **Ethernet Device** is used to communicate with a ControlLogix (Model 1756-L63) over the Ethernet network.

   To configure this, under Communication /Configure Driver, highlight the Ethernet Device driver (named

AB_ETH-1) and click **Configure**.

The PLC in this example has an IP address of **192.168.10.25**. You should see the IP address for the intended PLC in the Station Mapping grid similar to Figure 1 (below):

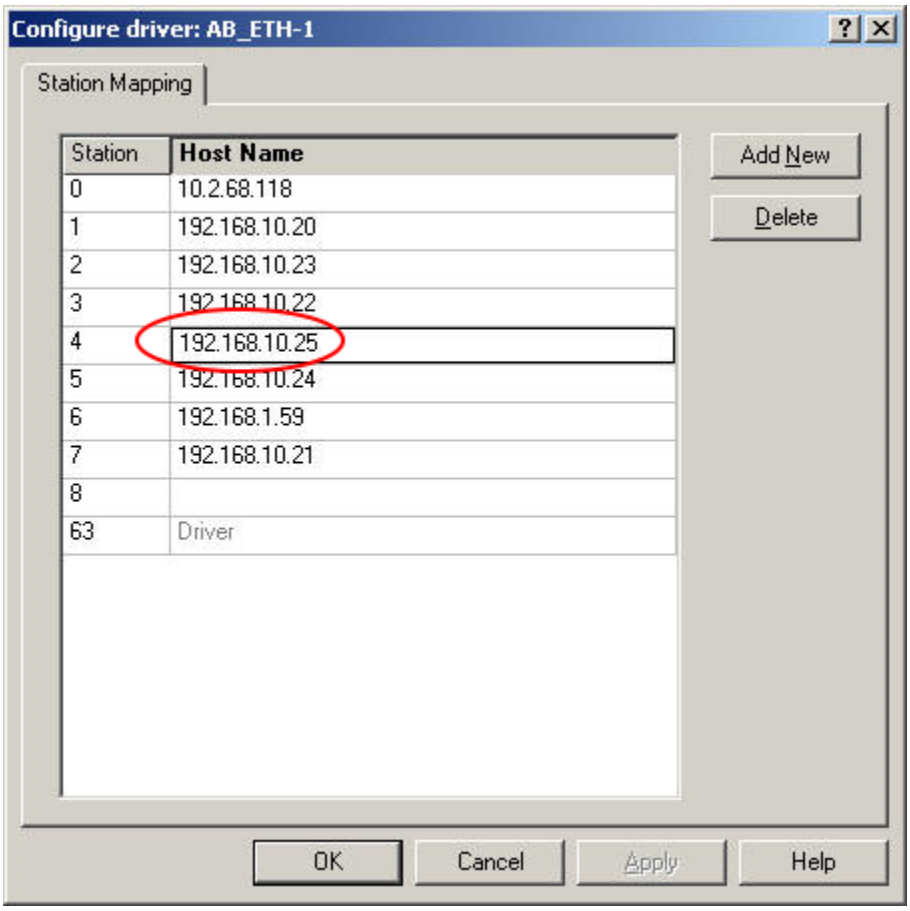


Figure 1: PLC Station Mapping

3. Click **OK** to close the Configure driver dialog box.

The **Configure Drivers** dialog box should show of the driver is running:

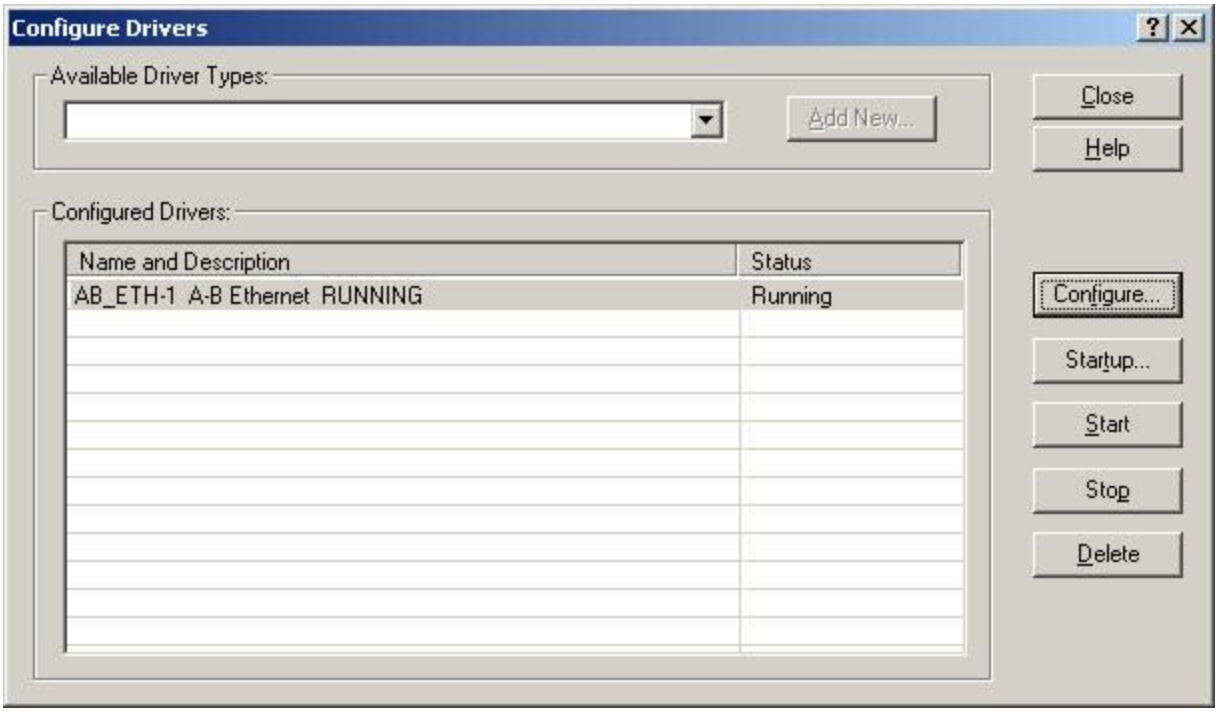


Figure 2: RSLinx Ethernet Device Communication Driver

4. Verify that RSLinx recognizes the PLC on the Ethernet network by using **Communications/RSWho.**

   Figure 3 (below) indicates that RSLinx can browse the PLC on the Ethernet network:

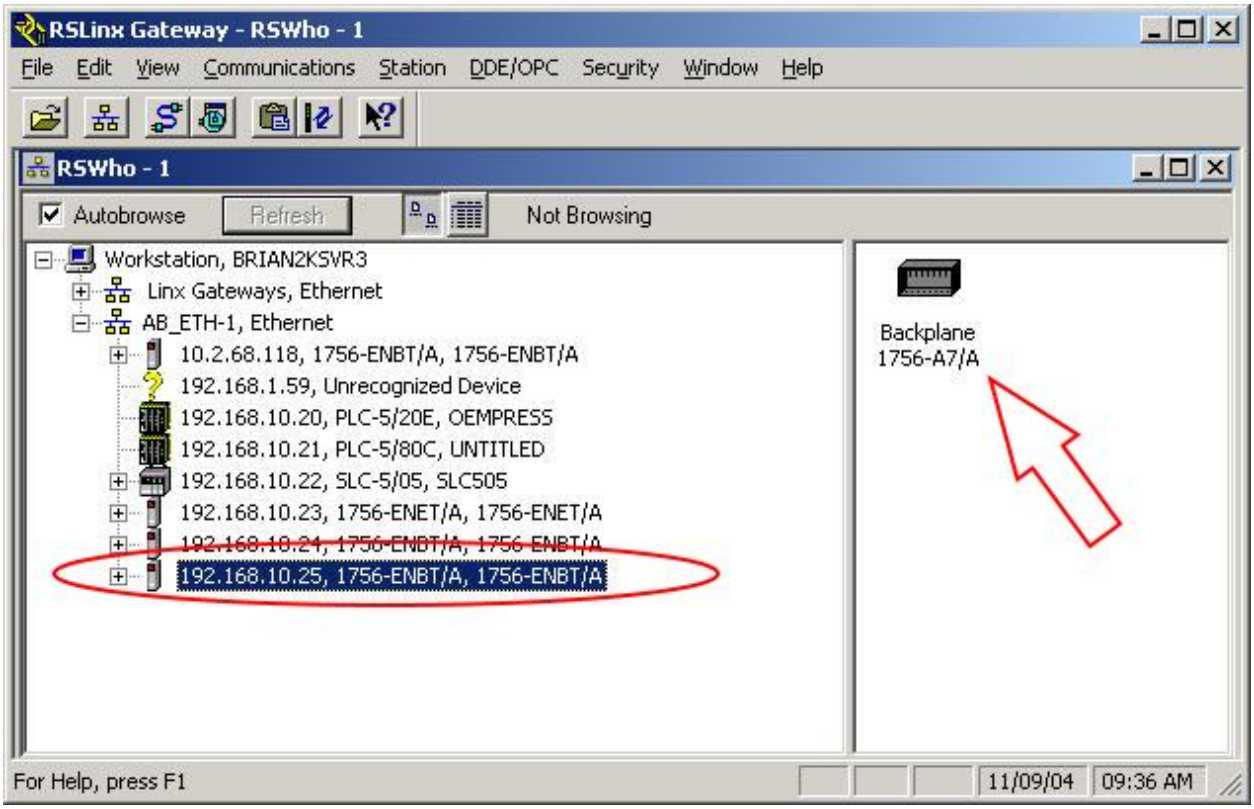


Figure 3: RSWho Found the PLC

5. Select **DDE/OPC /Topic Configuration** from the main menu .
6. Highlight the topic (Topic CLx5563) for the controller.

The respective PLC processor should also be highlighted on the right pane as shown in Figure 4 (below):

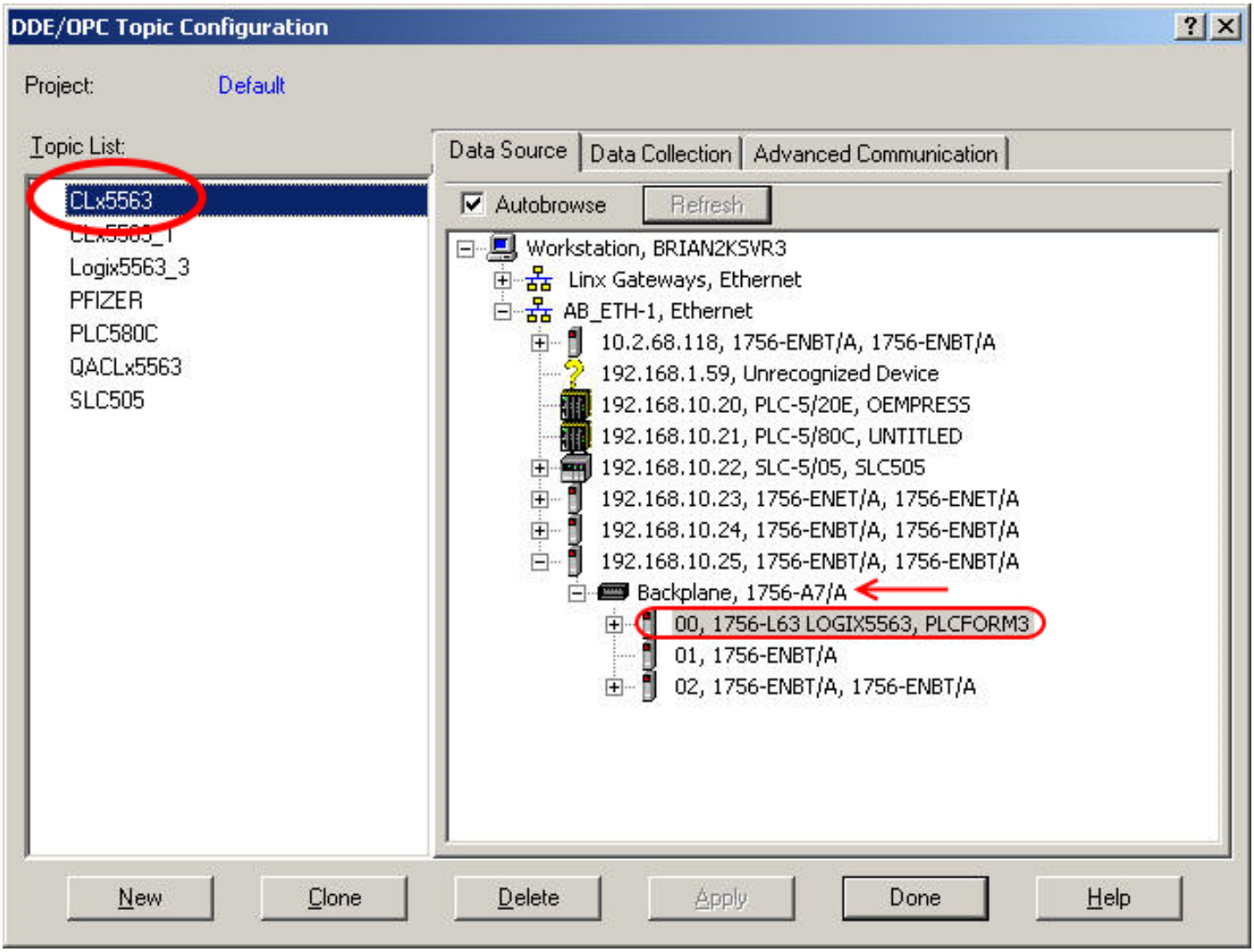


Figure 4: RSLinx DDE/OPC Topic Configuration

7. Click **Done** to close the Topic Configuration window.

Configure FSGateway

1. Select **Start/Programs/Wonderware/System Management Console** to launch the **System Management Console** (SMC).

2. From the System Management Console, navigate in the DAServer Manager to the FSGateway hierarchy (expand **DAServer Manager/Default Group/ Local/ArchestrA.FSGateway.1**).

See figure 5 (below):

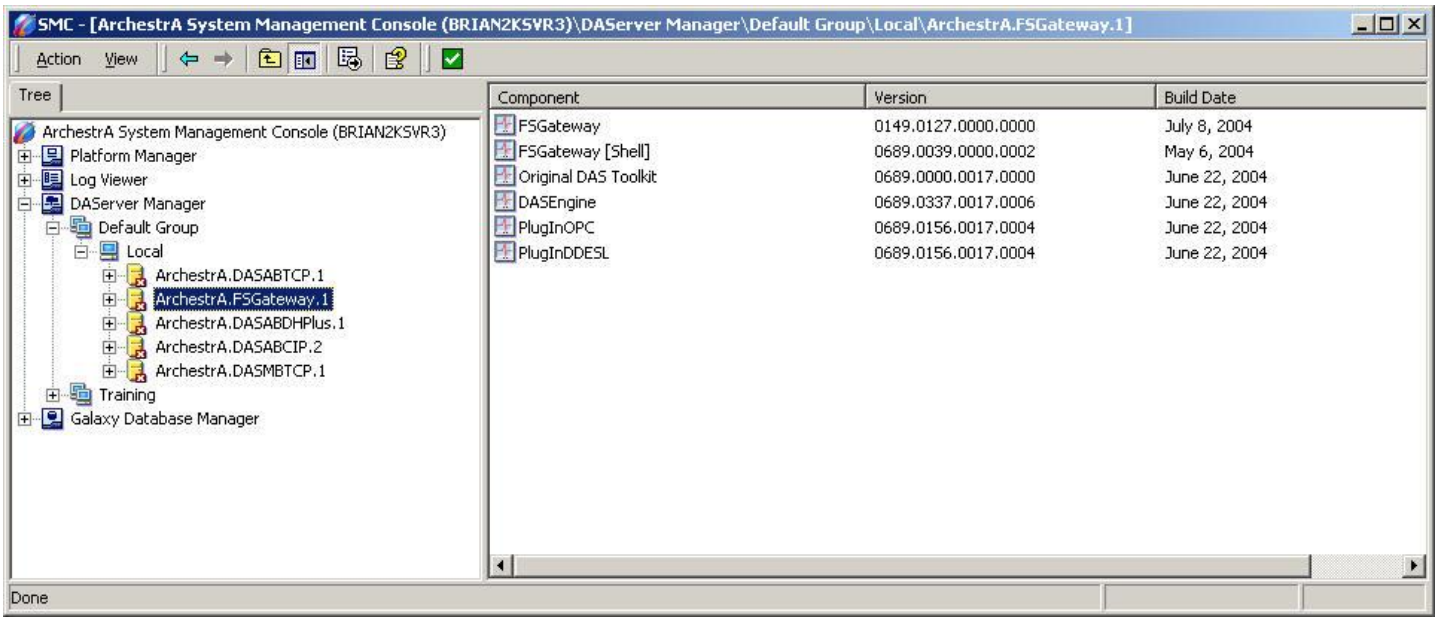


Figure 5: FSGateway in the SMC

3. Expand the **ArchestrA.FSGateway.1** icon.

4. Click on **Configuration object**. The Global Parameters dialog box will appear on the right pane of the window. See Figure 6 (below):

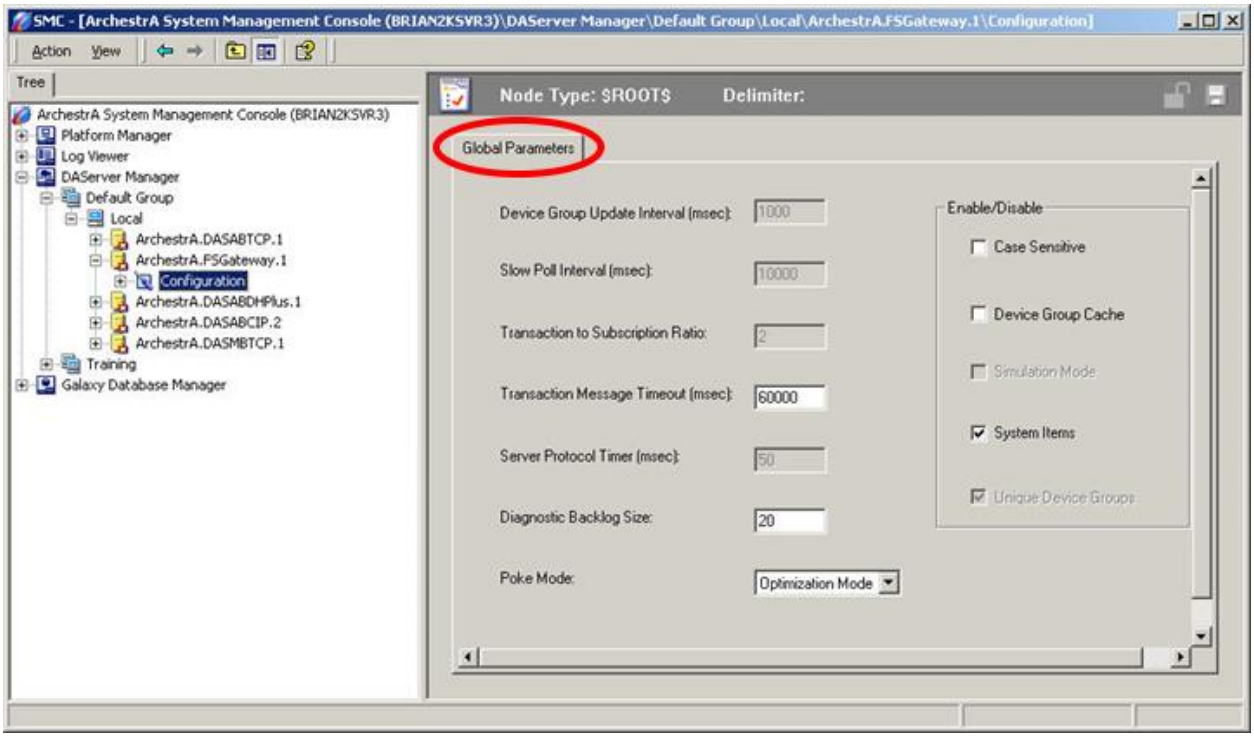
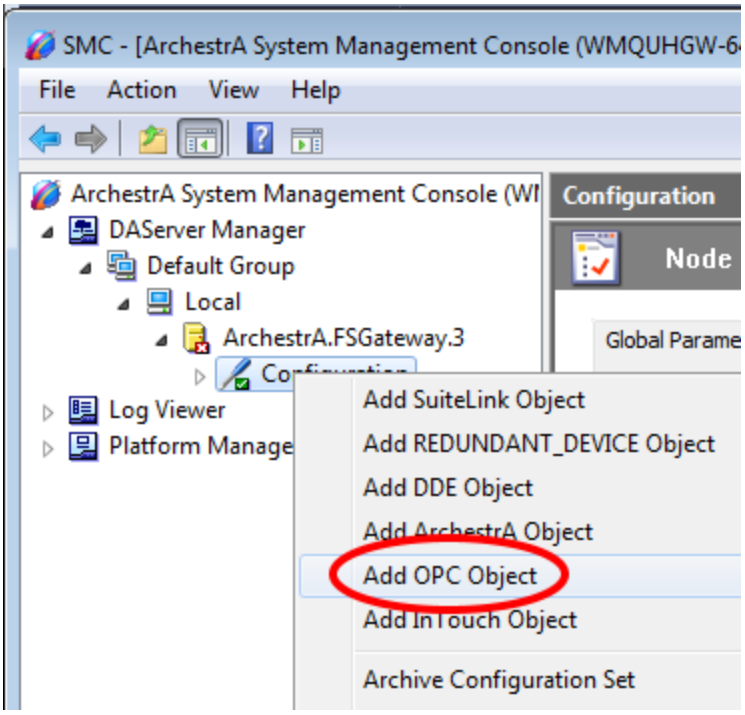


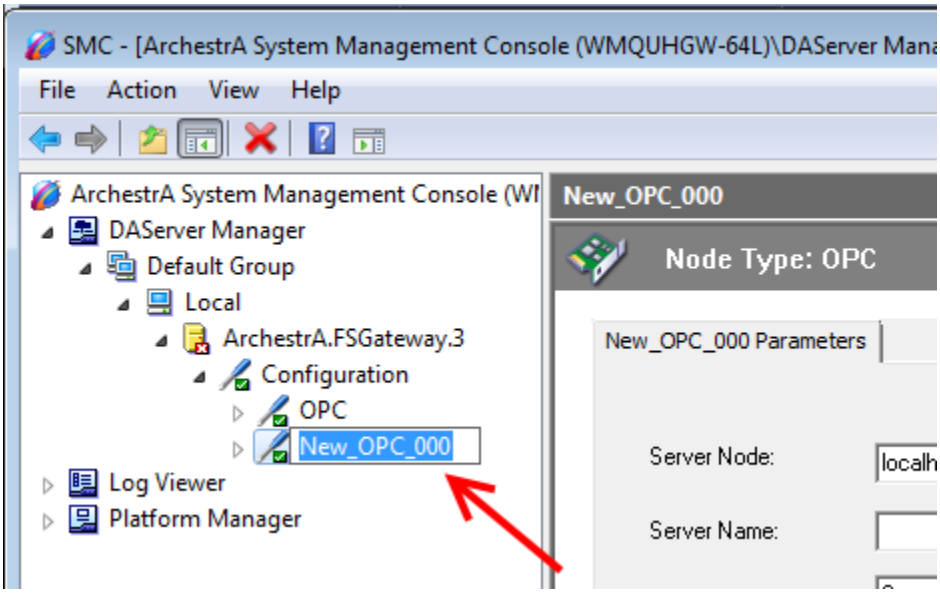
Figure 6: FSGateway Global Parameters

Note If the system has more than 5,000 items on advise, it is recommended that the Transaction Message Timeout should be set to 120 seconds.

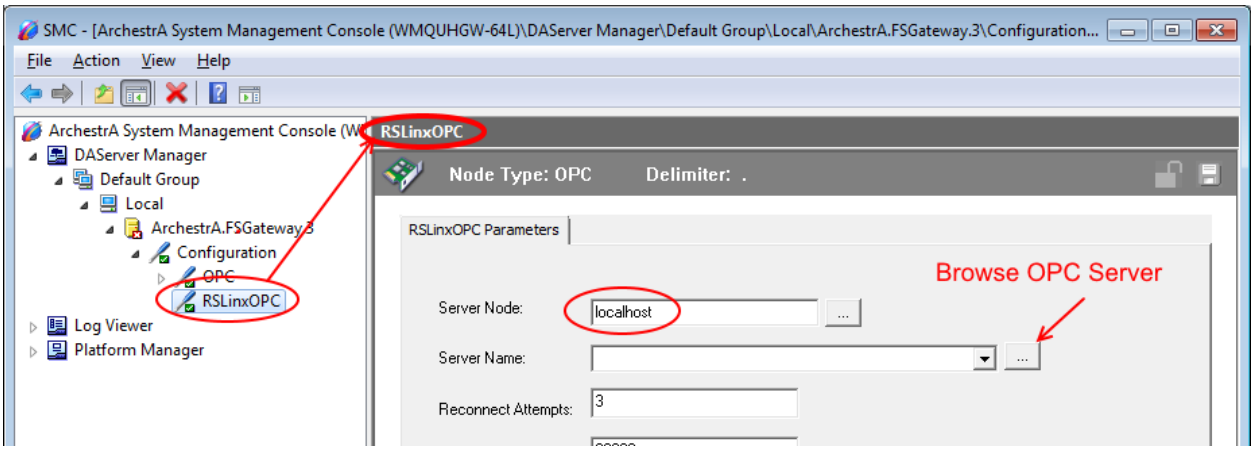
5. Right click the **Configuration** object and select **Add OPC Object**.



A new default **New_OPC_000** object is added to the hierarchy tree.



6. Right click **New_OPC_000** and select **Rename** to change it to a meaningful name such as **RSLinxOPC**.
7. Accept the default Server Node name of **localhost** (On the **OPC Parameters** window **RSLinxOPC Parameters** area , which indicates that RSLinx is running on the same computer with FSGateway.



Note: If RSLinx and FSGateway are on different computers, RSLinx Gateway version is required for remote connection. **If RSLinx and FSGateway are on different computers, the server node name should be the name of the computer running RSLinx.**

8. Click on the browse button to browse the OPC server, then select the **RSLinx** OPC Server.

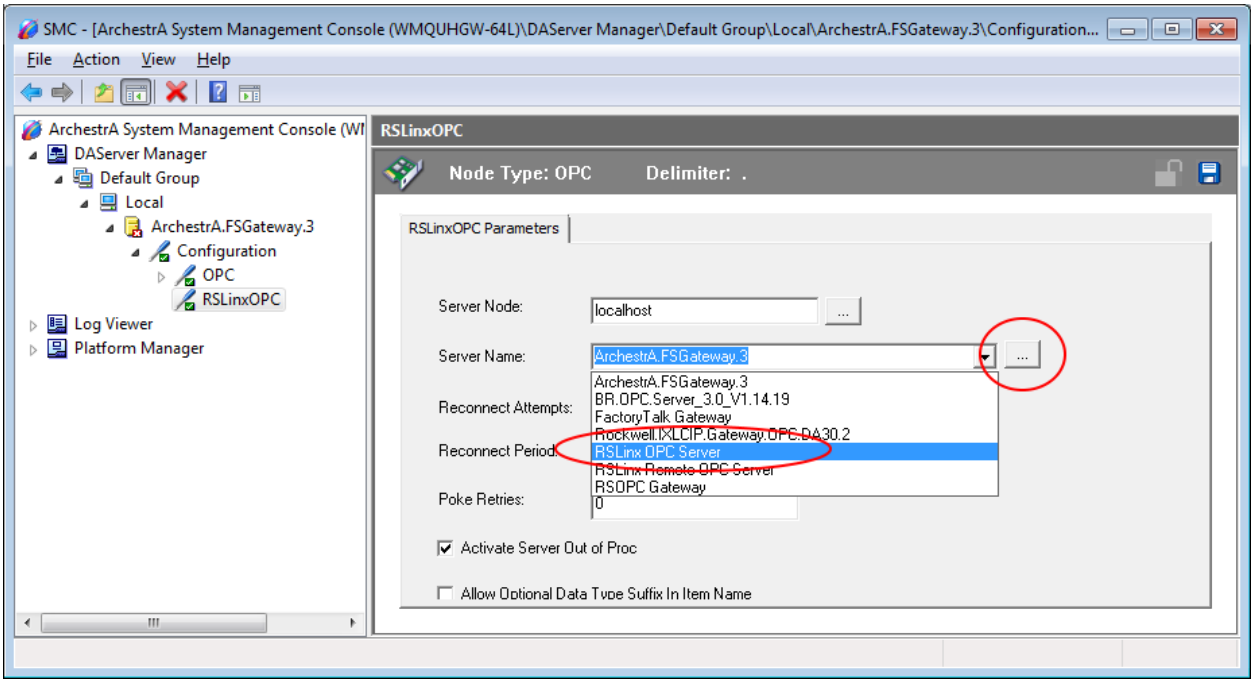


Figure 7: Selecting the RSLinx OPC Server

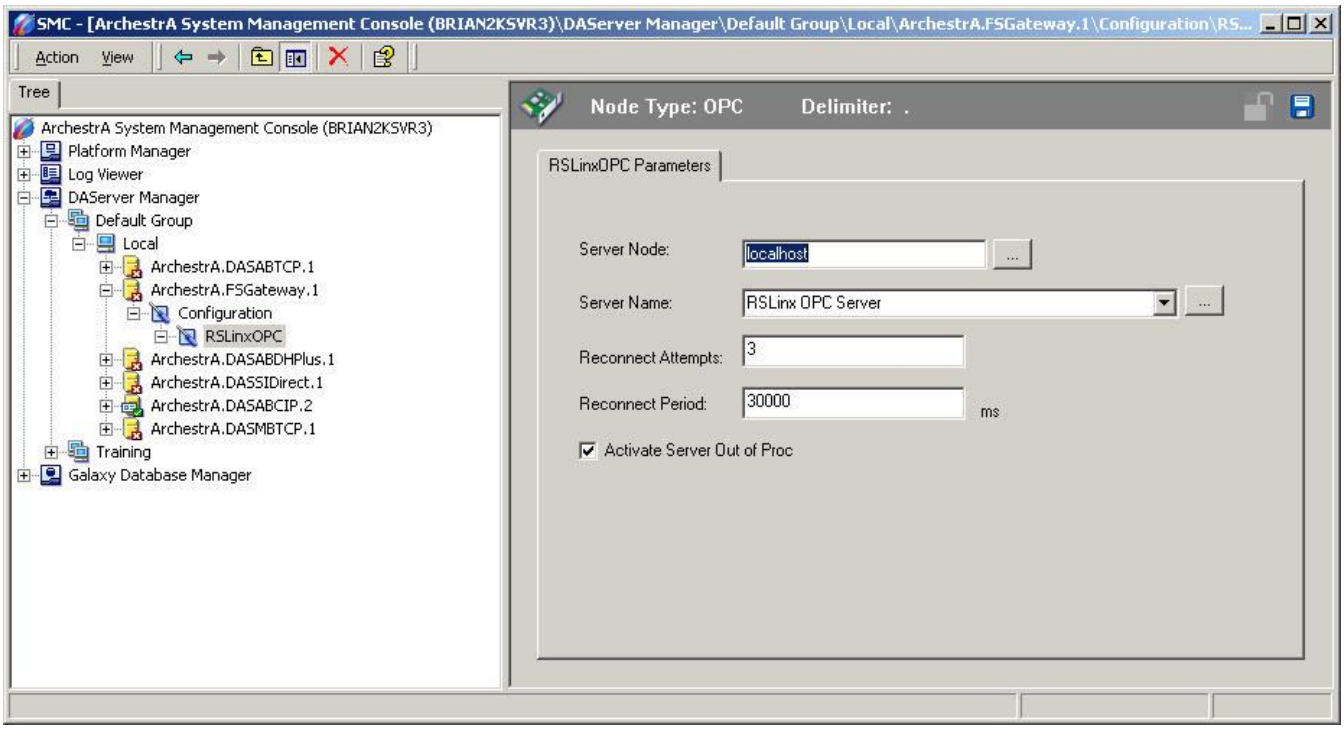
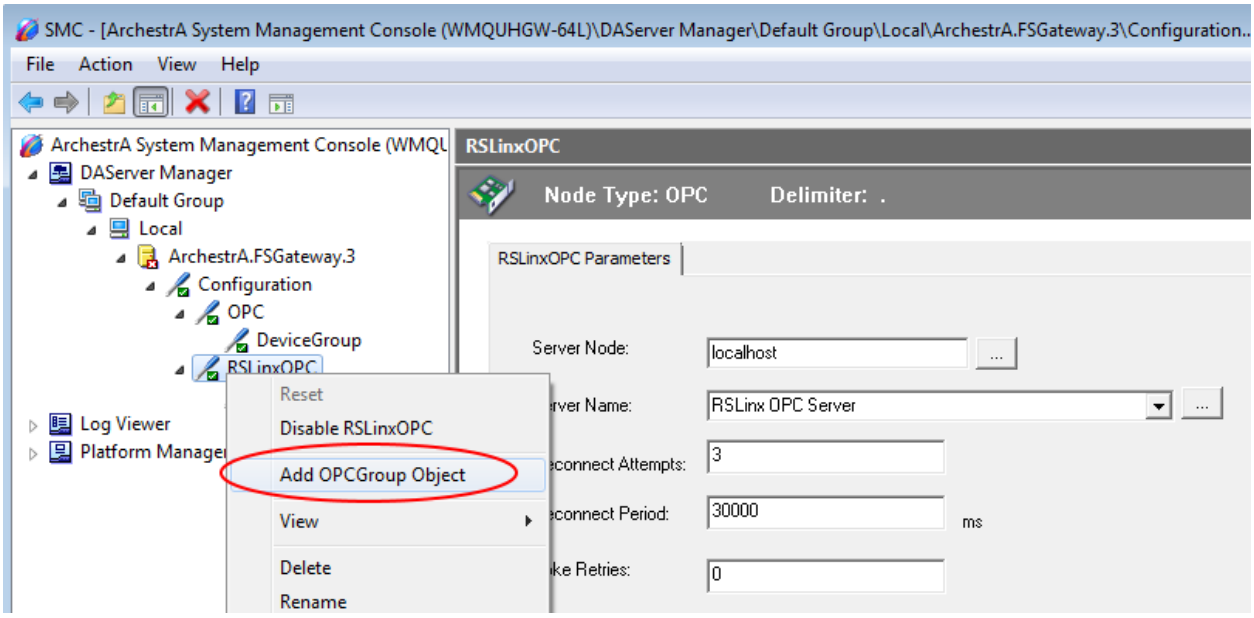


Figure 8: OPC Object Parameters

9. Right-click **RSlinxOPC** to add a new OPC group object **New_OPC_Group_000**.



After every change, if you haven't saved the configuration, you will be prompted to save it.

10. Click **Yes** to save the changes:

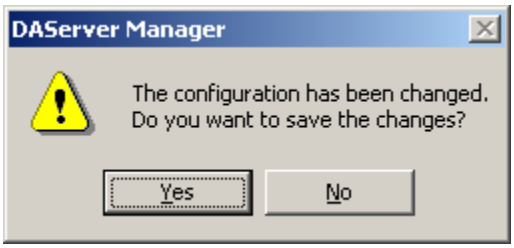


Figure 9: Save Prompt

11. Rename the default New_OPC_Group_000 to a meaningful name such as **CLX5563** (Figure 10 below):

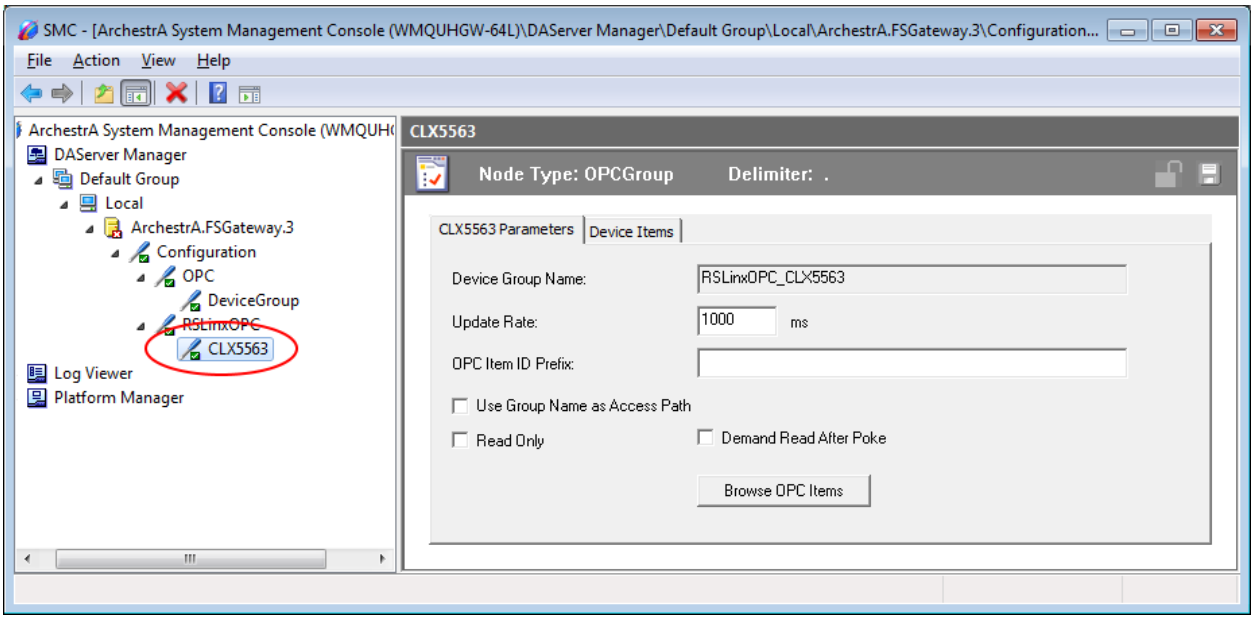


Figure 10: OPC Group Parameters

12. Click the **Browse OPC Items** button .

    The **OPC Item Browser** appears.

13. Select and expand the topic (**CLX5563**) defined in RSLinx as shown in the above step:

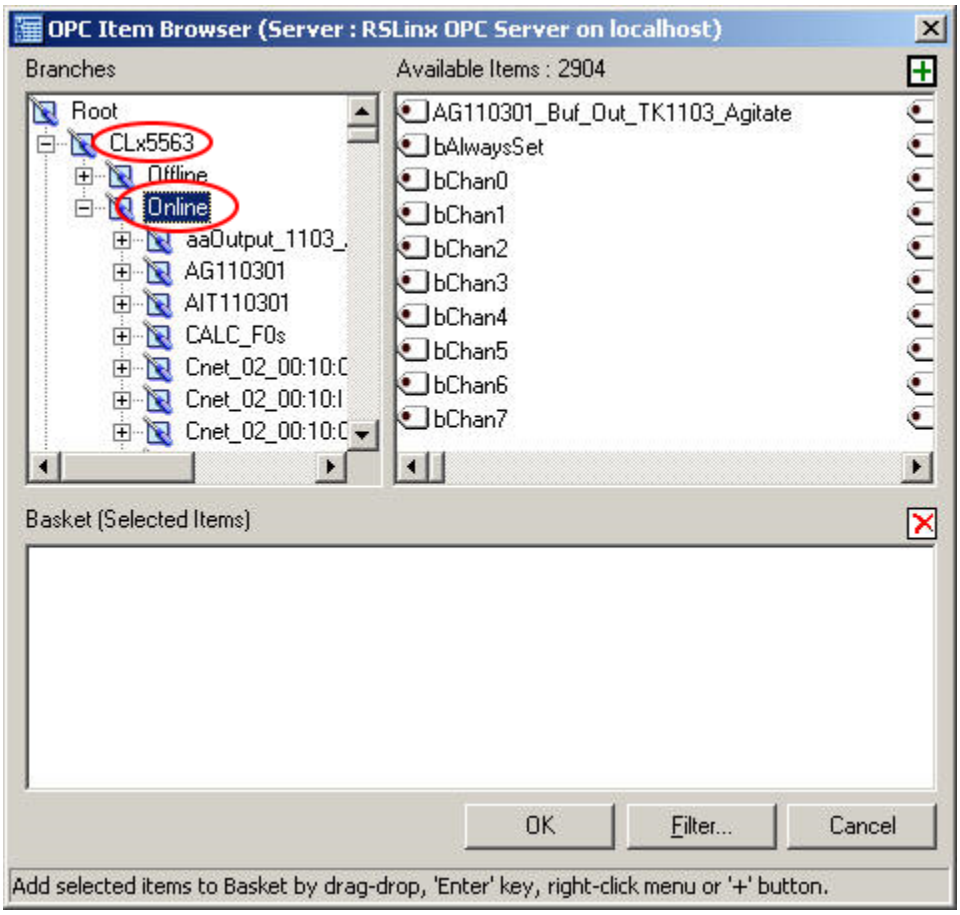


Figure 11: OPC Item Browser - Select the OPC group (RSLinx topic)

14. Expand the **Online** folder and select items in the PLC.

Since the PLC tags **MyBool**, **MyFloat**, and **MyInt** are defined as arrays of 10 elements each, we will add only the first element for each tag.

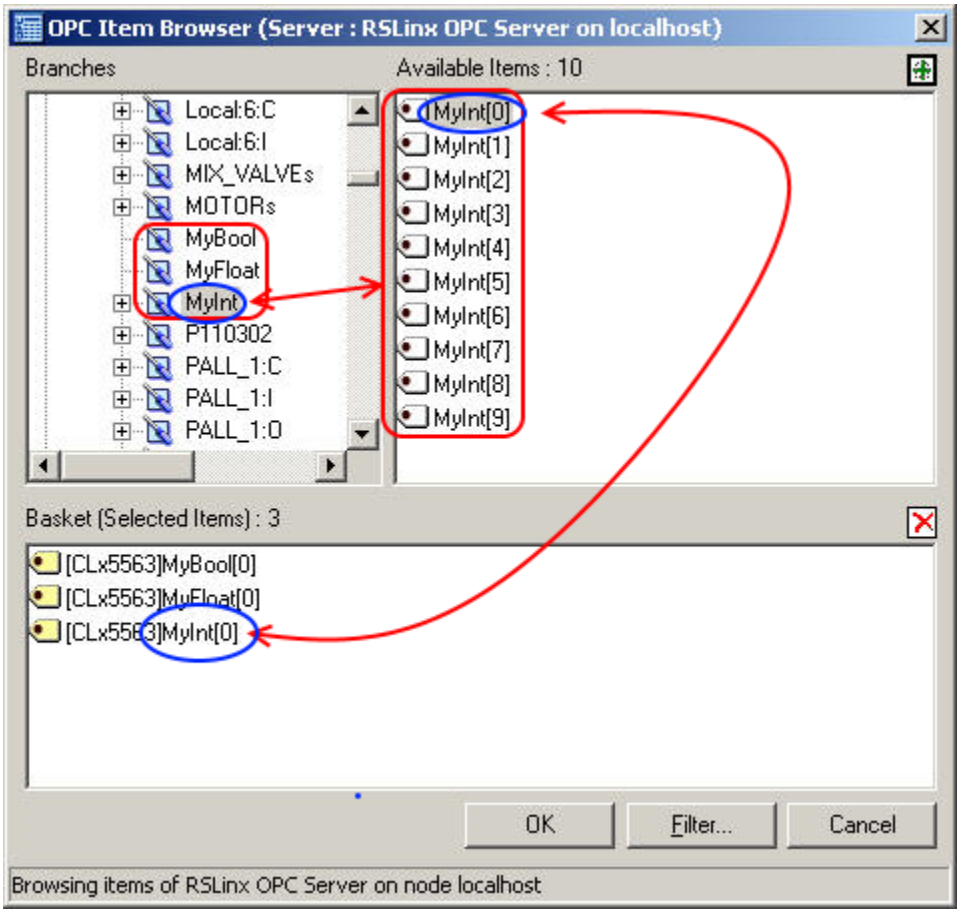


Figure 12: Add PLC tags

15. Click **OK** to return to the OPC Group Parameters tab field as shown in Figure 10 above.
16. Select the **Device Item** tab to display all the items we just added:

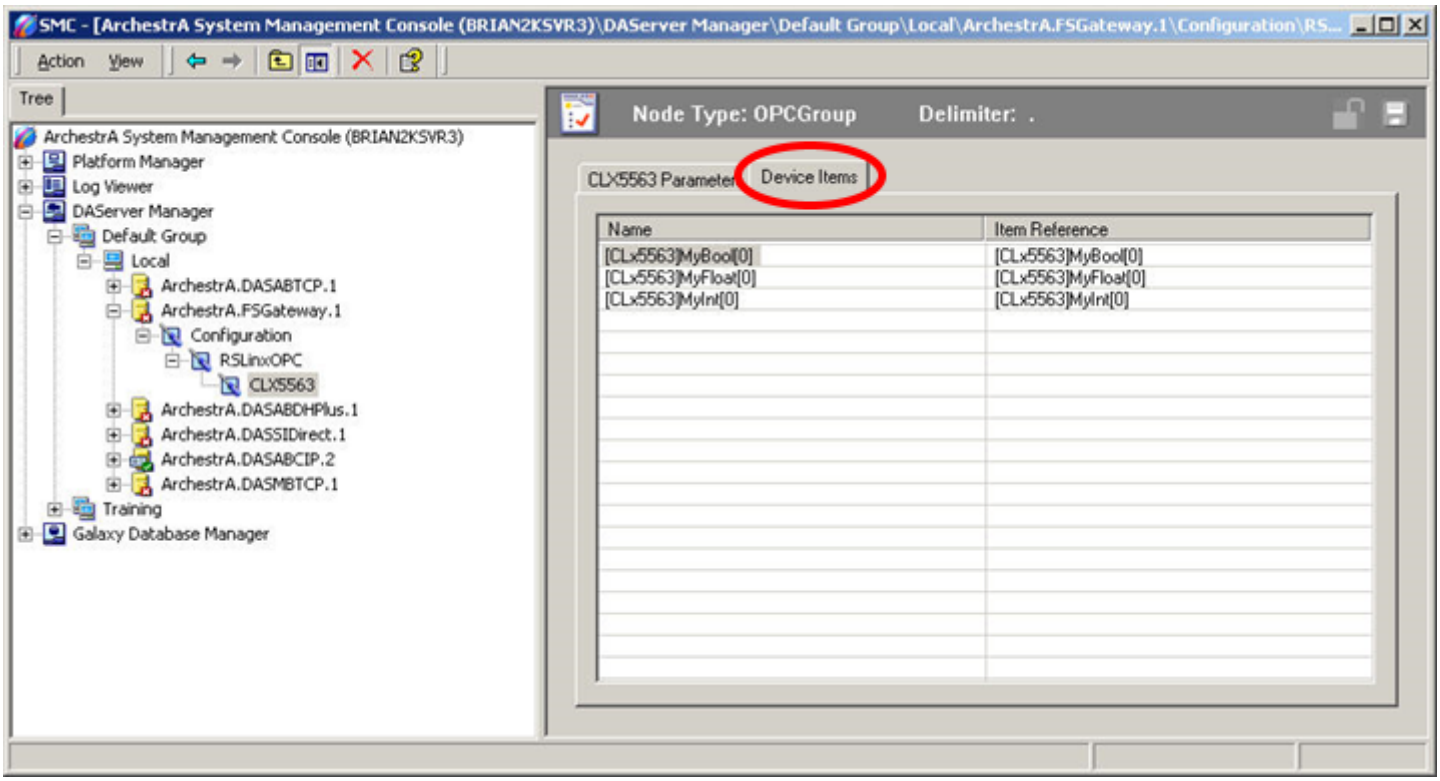


Figure 13: Device Items

The Device Items table includes two columns: **Name** and **Item Reference**. **This is where aliases are assigned for each item reference**.

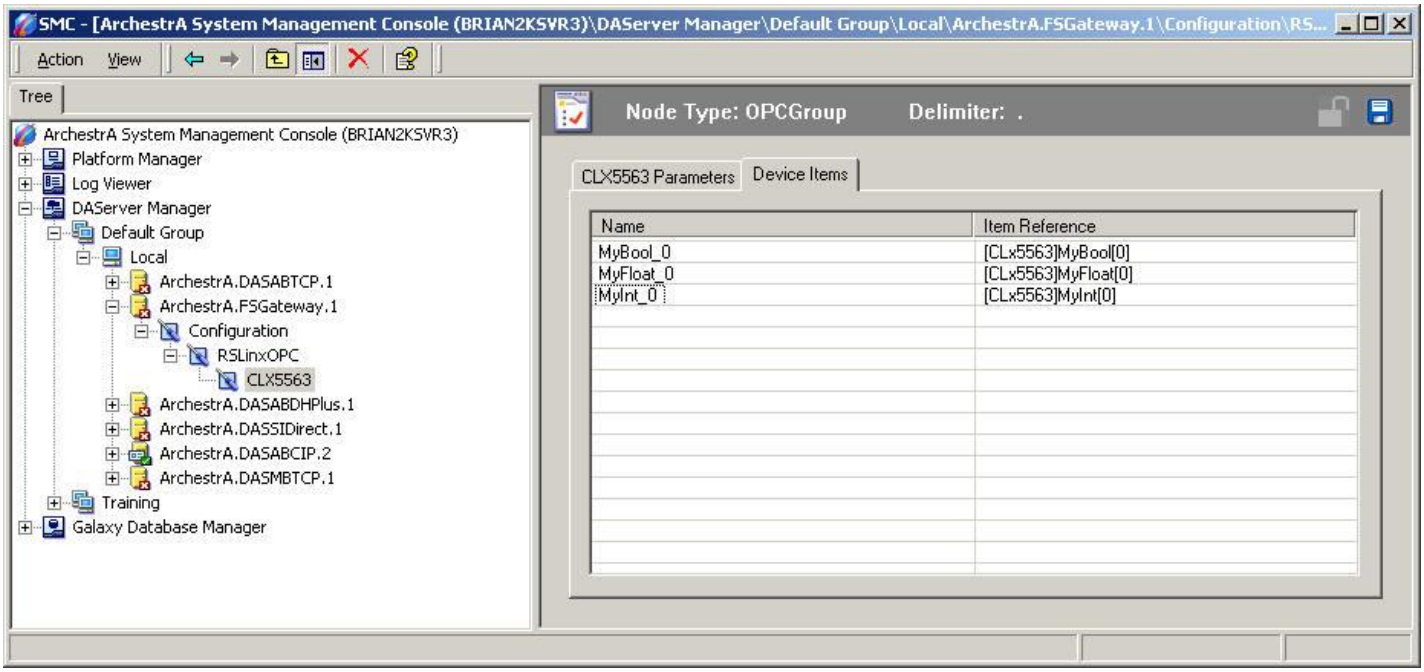


Figure 14: Device Item Aliases

The FSGateway is now ready for use. In order to use it, you must activate it.

17. Right-click **ArchestrA.FSGateway.1** and click on **Activate Server** on the shortcut menu.

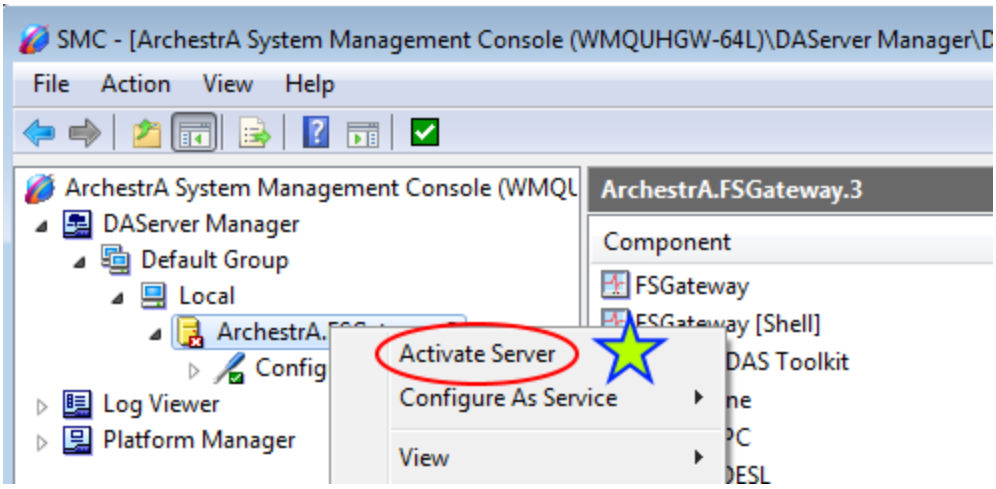


Figure 15: Activate the FSGateway

Test Communications

You can now test the connections to the PLC. We will use the **WWClient** utility for the test (click here to download).

To launch the WWClient:

1. Click **Start/Run** from the Windows taskbar.
2. Enter **WWClient** to launch the Wonderware WWClient program.
3. Select **Connections/Create** from the main menu bar.

   The **Create Connection** dialog box appears.
4. Enter appropriate information as shown in the following figure:

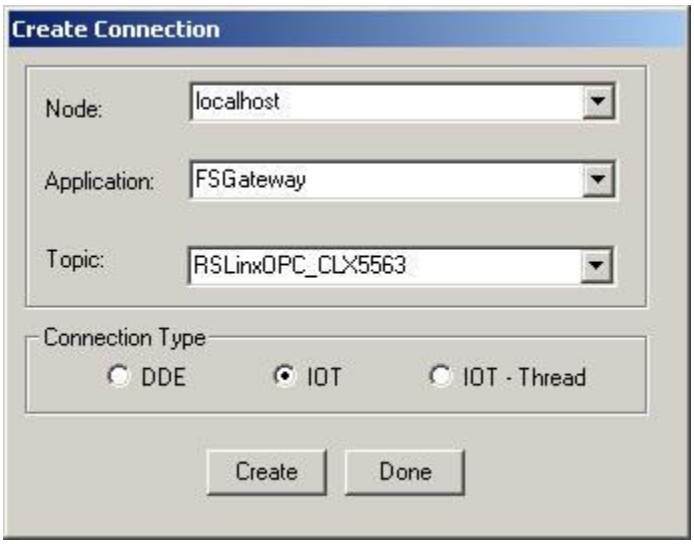


Figure 16: WWClient - Create Connection Dialog Box

5. Select **Item** on the main menu.
6. Enter a known good PLC tag in the Item entry box, and click **AdviseEX**.

In this example, alias items **MyBool_0**, **MyFloat_0**, and **MyInt_0** are entered:

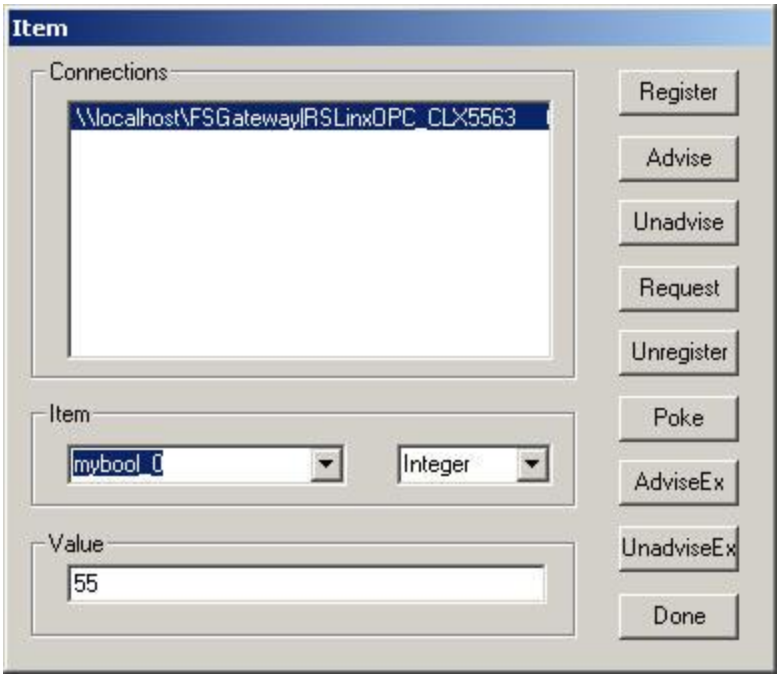


Figure 17: Advising tags

The following figure (Figure 18 below) shows an example of WWClient successfully advising items **MyBool_0**, **MyFloat_0**, and **MyInt_0**:

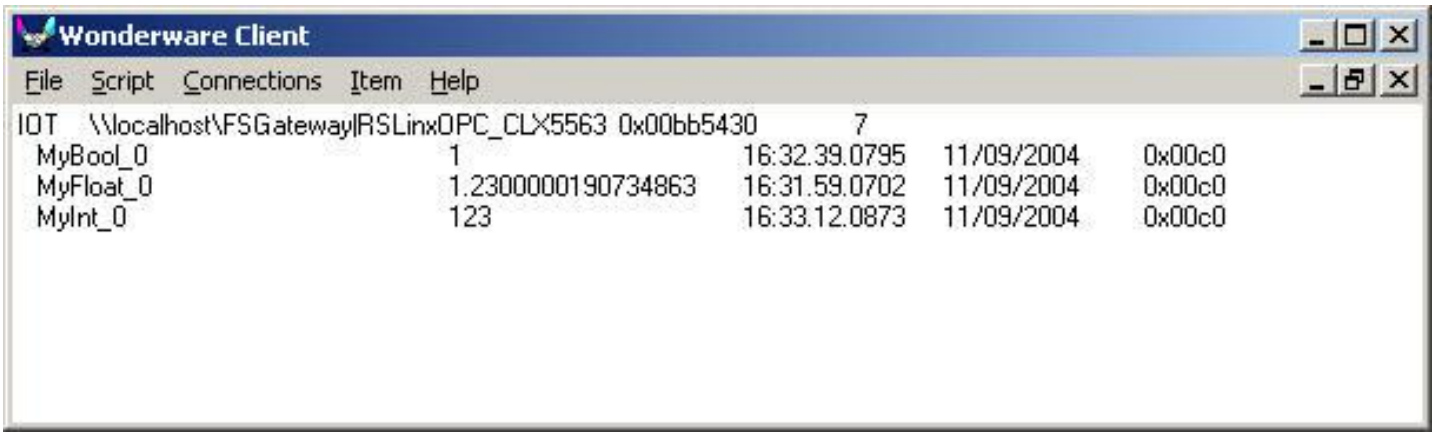


Figure 18: Successful Advise

DDE Examples

This section provides several examples of using DDE with SAS under Windows. These examples use Microsoft Excel and Microsoft Word as DDE servers, but any application that supports DDE as a server can communicate with SAS.

Before you run these examples, you must first invoke Microsoft Excel and Microsoft Word, and open the spreadsheet or document used in the example.

Note: DDE examples are included in the host-specific sample programs that you access from the Help menu. ■

Using the X Command to Open a DDE Server

A DDE server application can be opened using the X command within SAS code. The XWAIT and XSYNC options must be turned off.

```
options noxwait noxsync;
x '"c:\microsoft office\office\excel.exe"';
```

Double quotation marks are required around the path if the path contains a space. The single quotation marks are for the X command.

Using DDE to Write Data to Microsoft Excel

The first example sends data from a SAS session to an Excel spreadsheet. The target cells are rows 1 through 100 and columns 1 through 3. To send the data, submit the following program:

```
/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 100 and columns 1 through 3 */
filename random dde
   'excel|sheet1!r1c1:r100c3';
data random;
   file random;
   do i=1 to 100;
      x=ranuni(i);
      y=10+x;
      z=x-10;
      put x y z;
   end;
run;
```

Using DDE to Write Data to Microsoft Word

This example sends a text string to a Microsoft Word document at a given bookmark. Note the difference between using DDE with Microsoft Word and Microsoft Excel.

```
filename testit dde 'winword|"c:\temp\testing.doc"
                     !MARK' notab;

data _null_;
   file testit;
   put 'This is a test.';
run;
```

Note: If you are writing to Microsoft Word97, use Visual Basic commands such as FileOpen.Name, FileSave, FileClose, and Insert. If the PUT statement contains a macro that Word97 does not understand, you will see this message:

```
Ambiguous name detected: TmpDDE
```

Using DDE to Read Data from Microsoft Excel

You can also use DDE to read data from an Excel application into SAS, as in the following example:

```
/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 10 and columns 1 through 3  */
filename monthly
  dde 'excel|sheet1!r1c1:r10c3';
data monthly;
   infile monthly;
   input var1 var2 var3;
run;
proc print;
run;
```

Using DDE to Read Data from Microsoft Word

This example reads data from a Microsoft Word document at a given bookmark.

```
filename testit dde 'winword|"c:\temp\testing.doc"
                     !MARK' notab;

libname workdir 'c:\temp';

/* Get ready to read the first bookmark.  */

data workdir.worddata;
   length wordnum $5;
   infile testit;
   input wordnum $;
run;
```

```
proc print;
run;
```

Using DDE and the SYSTEM Topic to Invoke Commands in an Application Using Excel

You can issue commands to Excel or other DDE-compatible programs directly from SAS using DDE. In the following example, the Excel application is invoked using the X command; a spreadsheet called SHEET1 is loaded; data are sent from SAS to Excel for row 1, column 1 to row 20, column 3; and the commands required to select a data range and sort the data are issued. The spreadsheet is then saved and the Excel application is terminated.

```
/* This code assumes that Excel       */
/* is installed on the current        */
/* drive in a directory called EXCEL. */

options noxwait noxsync;
x '"c:\microsoft office\office\excel.exe"';

/* Sleep for 60 seconds to give */
/* Excel time to come up.       */

data _null_;
   x=sleep(60);
run;

/* The DDE link is established using   */
/* Microsoft Excel SHEET1, rows 1      */
/* through 20 and columns 1 through 3  */

filename data
   dde 'excel|sheet1!r1c1:r20c3';
data one;
   file data;
   do i=1 to 20;
      x=ranuni(i);
      y=x+10;
      z=x/2;
      put x y z;
   end;
run;

/* Microsoft defines the DDE topic */
/* SYSTEM to enable commands to be */
/* invoked within Excel.           */

filename cmds dde 'excel|system';

/* These PUT statements are       */
/* executing Excel macro commands */

data _null_;
   file cmds;
   put '[SELECT("R1C1:R20C3")]';
   put '[SORT(1,"R1C1",1)]';
   put '[SAVE()]';
   put '[QUIT()]';
run;
```

Using the NOTAB Option with DDE

SAS expects to see a TAB character placed between each variable that is communicated across the DDE link. Similarly, SAS places a TAB character between variables when data are transmitted across the link. When the NOTAB option is placed in a FILENAME statement that uses the DDE device-type keyword, SAS accepts character delimiters other than tabs between variables.

The NOTAB option can also be used to store full character strings, including embedded blanks, in a single spreadsheet cell. For example, if a link is established between SAS and the Excel application, and a SAS variable contains a character string with embedded blanks, each word of the character string is normally stored in a single cell. To store the entire string, including embedded blanks in a single cell, use the NOTAB option as in the following example:

```
/* Without the NOTAB option, column1  */
/* contains 'test' and column2        */
/* contains 'one'.                    */

filename test
  dde 'excel|sheet1!r1c1:r1c2';
data string;
   file test;
   a='test one';
   b='test two';
   put a $15. b $15.;
run;

/* You can use the NOTAB option to store   */
/* each variable in a separate cell. To    */
/* do this, you must force a tab           */
/* ('09'x) between each variable, as in    */
/* the PUT statement.                      */
/* After performing this DATA step, column1*/
/* contains 'test one' and column2         */
/* contains 'test two'.                    */

filename test
  dde 'excel|sheet1!r2c1:r2c2' notab;
data string;
   file test;
   a='test one';
   b='test two';
   put a $15. '09'x b $15.;
run;
```

Using the DDE HOTLINK

If the HOTLINK option is specified, the DDE link is activated every time the data in the specified spreadsheet range are updated. In addition, DDE enables you to poll the data when the HOTLINK option is specified to determine whether data within the range specified have been changed. If no data have changed, the HOTLINK option returns a record of 0 bytes. In the following example, row 1, column 1 of the spreadsheet SHEET1 contains the daily production total. Every time the value in this cell changes, SAS reads in the new value and outputs the observation to a data set. In this example, a second cell in row 5, column 1 is defined as a status field. Once the user completes data entry, typing any character in this field terminates the DDE link:

```
/* Enter data into Excel SHEET1 in   */
```

```
/* row 1 column 1. When you          */
/* are through entering data, place  */
/* any character in row 5            */
/* column 1, and the DDE link is     */
/* terminated.                       */

filename daily
  dde 'excel|sheet1!r1c1' hotlink;
filename status
  dde 'excel|sheet1!r5c1' hotlink;
data daily;
   infile status length=flag;
   input @;
   if flag ne 0 then stop;
   infile daily length=b;
   input @;

   /* If data have changed, then the */
   /* incoming record length         */
   /* is not equal to 0.             */

   if b ne 0 then
      do;
         input total $;
         put total=;
         output;
      end;
run;
```

It is possible to establish multiple DDE sessions. The previous example uses two separate DDE links. When the HOTLINK option is used and there are multiple cells referenced in the item specification, if any one of the cells changes, then all cells are transmitted.

Unless the HOTLINK option is specified, DDE is performed as a single one-time data transfer. The values currently stored in the spreadsheet cells at the time that the DDE is processed are values that are transferred.

Using the !DDE_FLUSH String to Transfer Data Dynamically

DDE also enables you to program when the DDE buffer is dumped during a DDE link. Normally, the data in the DDE buffer are transmitted when the DDE link is closed at the end of the DATA step. However, the special string '!DDE_FLUSH' issued in a PUT statement instructs SAS to dump the contents of the DDE buffer. This function allows you considerable flexibility in the way DDE is used, including the capacity to transfer data dynamically through the DATA step. The following example creates a Macro Sheet in Microsoft Excel. Commands are then written to the Macro Sheet, which will rename Sheet1 to NewSheet. After writing these commands, through the use of !DDE_FLUSH , the Excel Macro can be executed in the same DATA Step as it is written.

```
filename cmds dde 'excel|system';
data _null_;
file cmds;
/* Insert an Excel Macro Sheet */
put '[workbook.insert(3)]';
run;

/* Direct the Output to the Newly created Macro Sheet */
filename xlmacro dde 'excel|macro1!r1c1:r5c1' notab;
```

```
data _null_;
file xlmacro;
put '=workbook.name("sheet1","NewSheet")';
put '=halt(true)';
/* Dump the contents of the buffer, allowing us to both write and */
/* execute the macro in the same DATA Step                        */
put '!dde_flush';
file cmds;
/* Run Macro1 */
put '[run("macro1!r1c1")]';
put '[error(false)]';
/* delete the Macro Sheet */
put '[workbook.delete("macro1")]';
run;
```

Using Macro Variables to Issue DDE Commands

This example illustrates the use of a Macro Variable to issue a command to Microsoft Excel. In the example, the Macro Variable, excelOne, is being used in place of the Excel Workbook location C:\test.xls. Since macro triggers such as ampersands and percents are treated as text within single quotes, a Macro quoting function must be used. %STR is used to mask each individual apostrophe separately. Anytime you have an unmatched apostrophe or parenthesis then it must be preceded by a percent sign and since each apostrophe needs to be treated independently of each other the percents are needed. Once %STR has hidden the apostrophe the macro variable &excelOne resolves. %UNQUOTE is then used to remove what %STR has done and restores each apostrophe around the resolved value leaving the result as:

```
'[open("C:\test.xls")]'
options mprint symbolgen;
filename cmds dde 'excel|system';

%let excelOne=C:\test.xls;

data _null_;
file cmds;
put %unquote(%str(%'[open("&excelOne")]%'));
run;
```

Reading Missing Data

This example illustrates reading missing data from an Excel spreadsheet called SHEET1. This example reads the data in columns 1 through 3 and rows 10 through 20. Some of the data cells can be blank. Here is an example of what some of the data look like:

```
...
10   John    Raleigh      Cardinals
11   Jose    North Bend   Orioles
12   Kurt    Yelm         Red Sox
13   Brent                Dodgers
...
```

Here's the code that can read these data correctly into a SAS data set:

```
filename mydata
  dde 'excel|sheet1!r10c1:r20c3';
data in;
```

```
   infile mydata dlm='09'x notab
          dsd missover;
   informat name $10. town $char20.
            team $char20.;
   input name town team;
run;
proc print data=in;
run;
```

In this example, the NOTAB option tells SAS not to convert tabs that are sent from the Excel application into blanks. Therefore, the tab character can be used as the delimiter between data values. The DLM= option specifies the delimiter character, and '09'x is the hexadecimal representation of the tab character. The DSD option specifies that two consecutive delimiters represent a missing value. The default delimiter is a comma. For more information about the DSD option, see SAS Language Reference: Dictionary. The MISSOVER option prevents a SAS program from going to a new input line if it does not find values in the current line for all the INPUT statement variables. With the MISSOVER option, when an INPUT statement reaches the end of the current record, values that are expected but not found are set to missing.

The INFORMAT statement forces the DATA step to use modified list input, which is crucial to this example. If you do not use modified list input, you receive incorrect results. The necessity of using modified list input is not DDE specific. You would need it even if you were using data in a CARDS statement, whether your data were blank- or comma-delimited