CHAPTER 10

Controlling the SAS System from Another Application Using OLE

Introduction to Automating the SAS System

The SAS System can perform as an OLE automation server. This means that you can use an application that can act as an OLE automation controller (such as Visual Basic) to create a SAS session and control it using the methods and properties that the SAS System makes available.

Many Windows applications use Visual Basic or Visual Basic for Applications as the scripting language for automation. All examples that are provided in this document use Visual Basic, but you can achieve the same results with any application that can act as an OLE automation controller.

Creating an Instance of the SAS System

To create an instance of the SAS System (that is, invoke a SAS session), you must create an OLE object by using the SAS System program identifier as it is listed in the Windows registry. The SAS program identifier is **SAS.Application**. Here is a Visual Basic example that instantiates (creates an instance of) a SAS session:

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
```

This example sets the identifier **OleSAS** to the new SAS session. You can then use this identifier to access the methods and properties that the SAS System makes available.

If you want to control an existing SAS automation object by using OLE automation, you can use your automation controlling language. In Visual Basic, you can use the following:

```
Dim OleSAS as Object
Set OleSAS = GetObject(,"SAS.Automation")
```

Note that this code does not create an instance of SAS if one does not already exist. Also, the existing SAS session must have been created as an OLE automation object (for example, using **CreateObject** in Visual Basic). You cannot use OLE automation to control a SAS session you invoked by using another method (for example, by using the **Start** menu).

Getting Feedback from the SAS Session

The SAS System provides two properties, RC and ResultString, that make it possible to pass information from the SAS session that you are automating back to the application that is controlling it. RC can contain a number; ResultString can contain a text string.

To set the values of these properties from within the SAS session, use the SETRC function with this syntax:

error=SETRC("*result-string*", *rc-number*);

where *result-string* is the value to be assigned the ResultString property, and *rc-number* is the value to be assigned to the RC property.

For example, you can use the Submit method to submit DATA step code that returns an error code as part of its processing. You can then check the value of that error using the RC or ResultString property. Here is a Visual Basic example of this:

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Wait = True
OleSAS.Submit("data _null_;
       error=setrc('Error string', 2.0);
       put error; run;")
If (OleSAS.RC <> 0) Then
    Response = MsgBox(OleSAS.ResultString,
               vbOKOnly, "Error message is",
               0, 0)
EndIf
```

Note that the Wait property is set to True in this example. This ensures that the DATA step finishes processing before the automation controller checks the value of the RC property.

Examples of Automating SAS with OLE

The following examples use Visual Basic as the scripting language to control the SAS System with OLE automation. You can use any scripting language from any Windows application that can act as an OLE automation controller.

Creating a SAS Automation Object

This Visual Basic code defines an object and creates an instance of the SAS System to associate with that object.

```
Dim OleSAS As Object
Set OleSAS = CreateObject("SAS.Application")
```

Determine Whether the SAS Session is Busy

This Visual Basic code queries the SAS session (using the Busy property) to test whether the session is busy processing code.

```
If (OleSAS.Busy) Then
   Response = MsgBox("SAS Session is Busy",
              vbOKOnly, "SAS Session", 0, 0)
Else
   Response = MsgBox("SAS Session is Idle",
              vbOKOnly, "SAS Session", 0, 0)
End If
```

Toggle the SAS Session between Visible and Invisible

This Visual Basic code hides or unhides the SAS session based on its current state.

```
OleSAS.Visible = Not OleSAS.Visible
```

Set the Main SAS Window Title of the SAS Session

This Visual Basic code assigns a title to the main SAS window of the SAS session and then displays the title in a message box.

```
OleSAS.Title = "Automation Server"
Response = MsgBox(OleSAS.Title, vbOKOnly,
           "Title Is", 0, 0)
```

Assign a SAS Data Library and Run a SAS Procedure

This Visual Basic code submits SAS code to the SAS session, assigning a SAS data library and running the INSIGHT procedure on sample data.

```
OleSAS.Submit("libname insamp
                'c:\sas\insight\sample';
               proc insight data=insamp.drug;
               run;")
```

End the SAS Session

This Visual Basic code ends the SAS session provided that there are no other OLE automation controllers making use of it.

```
OleSAS.Quit
Set OleSAS = Nothing
```

Methods and Properties for Use with a SAS OLE Automation Object

Once instantiated, the SAS OLE automation object supports these methods as well as several properties:

- □ "Command Method" on page 212
- □ "QueryWindow Method" on page 212
- □ "Quit Method" on page 213
- □ "Submit Method" on page 213
- □ "Top Method" on page 214
- □ "Properties for Controlling a SAS Automation Object" on page 215

Command Method

Invokes a command as if it was entered from the SAS command line

Syntax

Command("*sas-command*")

Details

By default, the active window receives the command. You can change which window receives the command by changing the CommandWindow property.

Example

This Visual Basic code invokes a SAS session and opens the BUILD window:

```
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Command("build")
```

QueryWindow Method

Queries whether a specified window exists within the SAS session

Syntax

QueryWindow("*window-name*")

Details

QueryWindow returns either True or False based on whether the specified window is open in the automated SAS session. If the window exists but is not visible, QueryWindow still returns True.

The window name that you specify must match the exact spelling of the window name in SAS. The *window-name* argument is not case sensitive.

Example

This Visual Basic code gets an existing SAS session and checks whether the BUILD window is open. If the window is not open, this code invokes BUILD:

```
Dim OleSAS as Object
Set OleSAS = GetObject(,"SAS.Application")
If (Not OleSAS.QueryWindow("build")) Then
   OleSAS.Command("build")
EndIf
```

Quit Method

Ends the SAS session

Syntax

Quit

Details

If the automation controller that issues the Quit method is the only controller that is using that particular SAS session, then the SAS session ends. If at least one other automation process is still using the SAS session, then the session remains running.

If you do not use the Quit method at the end of your automation sequence and the Visible property for the SAS session is set to True, then the SAS session remains running and is available for user interaction. If the Visible property is not set to True, then the SAS session ends.

Submit Method

Submits DATA step or procedure code for processing

Syntax

Submit("*SAS-program-code*")

Details

The string of text that you specify as *SAS-program-code* can contain multiple SAS statements separated by semicolons. The contents of the string are submitted to the SAS System for processing.

Example

The following example references a data library and invokes a SAS/AF application:

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Visible = True
OleSAS.Submit("libname afapp 'f:\sas\afapp';")
OleSAS.Command("af c=afapp.bigapp.main.frame")
```

Top Method

Brings the SAS session to the foreground

Syntax

Top

Details

The Top method works only if the Visible property is set to True.

Example

The following example invokes a SAS session, makes it a visible window, and then brings it to the foreground.

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
OleSAS.Visible = True
OleSAS.Top
```

Properties for Controlling a SAS Automation Object

Specify various properties of the SAS automation object

Properties and Descriptions

Busy
: indicates whether SAS is idle or working (for example, running a procedure, DATA step, and so on). This property is read only.

CommandWindow
: sets the window (based on the window title) to receive commands you specify using the Command method. The name you specify must match the spelling of the window name exactly (though this property is not case sensitive). Once set, the window receives subsequent commands you specify with the Command method until CommandWindow is changed or set to Null (by specifying ""). If Null, which is the default, the currently active window receives the command. This property is read/write.

CommandWindowVisible
: controls whether the window specified by the CommandWindow property is visible. If set to False, the window specified by the CommandWindow property is set to invisible. If the CommandWindow property is Null, this property has no effect. This property is read/write.

ConfirmExit
: controls the behavior of how SAS exits. A value of 0 means that no confirmation box is displayed before SAS exits. A value of 1 means that a confirmation box is displayed before SAS exits. A value of 2 selects the default action, which is controlled by an alternative method that defines how SAS exits; for example, the Preferences dialog.

Height
: sets the height, in pixels, of the SAS application window. This property is read/write.

Parent
: sets the name of the parent window that contains the SAS application window. If you change this value to another window, the SAS application window resizes to fit in the new frame. This property is read/write.

RC
: returns the return code passed by a user function. You can set this property from within the SAS session by using the SETRC function. This property is read-only from the automation controller.

ResultString
: returns a string passed by a user function. You can set this property from within the SAS session by using the SETRC function. This property is read-only from the automation controller.

Title
: sets the main SAS window title. This property is read/write.

Visible
: controls whether SAS is visible. This property is read/write.

Width
: sets the width, in pixels, of the SAS application window. This property is read/write.

X
: sets the horizontal coordinate, in pixels, for the top left corner of the SAS application window. This property is read/write.

Y
: sets the vertical coordinate, in pixels, for the top left corner of the SAS application window. This property is read/write.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. pp.555.

SAS Companion for the Microsoft Windows Environment, Version 8

ISBN 1–58025–524–8

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999