Chapter 83
OLE Classes

Overview

The OLE classes enable you to include an Object Linking and Embedding (OLE) object in a FRAME entry. There are three types of OLE classes: the OLE Insert class, the OLE Paste class, and the OLE Read class. Each class contains the same standard methods and behavior; they are distinguished primarily by the procedure you use to insert them into a FRAME entry. Each type of OLE object also has its own set of behaviors, which you can access through the SAS/AF methods described here.

Note: The OLE classes are available only on operating systems that support OLE, such as Microsoft Windows. For complete information about using OLE with the SAS System, see your host companion.

Each OLE class has its own item on the object Selection List:

- When you select OLE - Insert Object, you use the Microsoft Insert Object Dialog to select the OLE object.
- When you select OLE - Paste Special, you use the Microsoft Paste Special dialog to select the OLE object. (Note that you must have an object stored in the Windows Clipboard before you can paste it into a FRAME entry.)
- When you select OLE - Read Object, you use the OLE - Read Object Attribute window to specify an existing HSERVICE entry in a SAS catalog. No intermediate dialog is used for the OLE Read class.

For more information on the Microsoft dialogs, click Help for the dialog. After you select the object to include, you use the OLE Object Attribute window to specify the object’s attributes in the FRAME entry.

For complete information about how to include OLE objects in a FRAME entry, see your host companion.

When you include an OLE object with Paste Special or Insert Object, you can choose whether to embed the object in your frame or create a link to the object’s source. An embedded object stores all of its information within the document; it remains unchanged unless you edit it from within the frame. A linked object is not stored in the document, but is a link to the information stored elsewhere; you can edit this information outside of the FRAME entry, and the object will reflect the changes. If the location of the linked information changes, the link must be updated to reflect the new location (using the OLE Links dialog). Also, if you distribute the SAS/AF application that uses the linked object, you must distribute the linked information as well.
OLE controls (also called OCXs) are a special type of OLE object with these additional features:

- They provide a property page interface that enables a user to manipulate the control.
- They can generate events, which you can then use to drive FRAME, PROGRAM, and SCL entries. Events are the way an OLE control communicates information back to the FRAME entry.

For examples of how to use OLE custom controls, see your host companion.

Parent:
  sashelp.fsp.system.class

CLASSES:
  sashelp.fsp.Insert.class
  sashelp.fsp.Paste.class
  sashelp.fsp.ReadOLE.class

For more information on these classes, see:
Methods” Methods” on page 1532

---

**Methods**

Methods specific to the OLE class are described here. Inherited methods are described in the Widget class.

---

**Dictionary**

---

**_compute**

Invokes an OLE-supplied method that returns a value

**Syntax**

CALL NOTIFY (OLE-object, '_compute', in-OLE-method<, in-parm,...in-parm>,
  out-value);
The _compute method sends an OLE-supplied method and its parameters to an OLE object. The number of parameters (in-parm arguments) needed varies among different OLE objects and methods. Only OLE-supplied methods that have a return value should be used with the _compute method. For OLE methods with no return values, use the _do method.

Example

The following example returns the item in position 2 of an OLE control, storing the result in the character variable item2obj:

```
length item2obj $ 200;
call notify('oleobj', '_compute', 'GetItem', 2, item2obj);
```

The following example sets the cell value for row 2, column 5 of a spreadsheet OLE automation object to 100:

```
call send(oleobj, '_compute', 'Cells', 2, 5, cellobj1);
call send(cellobj1, '_setProperty', 'Value', 100);
```
_do

Invokes an OLE-supplied method that has no return value

Syntax

CALL NOTIFY (OLE-object, '_do', in-OLE-method<, in-parm,...in-parm>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-OLE-method</td>
<td>C</td>
<td>specifies the OLE-supplied method name</td>
</tr>
<tr>
<td>in-parm</td>
<td>C</td>
<td>N</td>
</tr>
</tbody>
</table>

Details

The _do method sends an OLE-supplied method and its parameters to an OLE object. The number of parameters (in-parm arguments) needed varies among different OLE objects and methods. Only OLE-supplied methods that have no return value should be used with the _do method. For OLE-supplied methods with return values, use the _compute method.

Example

The following example sends the 'AboutBox' method to an OLE control, which displays the About Box for the control:

```plaintext
call notify('oleobj', '_do', 'AboutBox');
```

-enableDefaultAction

Enables the OLE object's default action

Syntax

CALL NOTIFY (OLE-object, '_enableDefaultAction');

Details

This method enables the default verb for an OLE object to execute when the object is double clicked. By default, the default action is enabled.
_execute

Executes an OLE verb for the object

**Syntax**

CALL NOTIFY (OLE-object, '_execute', in-verb<, in-verb,...in-verb>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-verb</td>
<td>C</td>
<td>specifies the OLE verb to execute</td>
</tr>
</tbody>
</table>

**Details**

A list of verbs supported by this object can be viewed in the Associated Verbs window for the object. You can specify more than one verb in a single call to _execute.

If you attempt to execute a verb that is not valid for the object, the SCL program halts and returns a message that the verb does not exist.

_getEvent

Returns the name of the last OCX event received

**Syntax**

CALL NOTIFY (OLE-object, '_getEvent', out-event);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-event</td>
<td>C</td>
<td>contains the returned name of the last OCX event received</td>
</tr>
</tbody>
</table>

**Details**

This method returns the name of the event that the specified OLE object most recently received.

_getProperty

Returns the value of a property of an OLE object
**Syntax**

CALL NOTIFY (OLE-object, '_getProperty', in-OLE-property, out-property-value);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-OLE-property</td>
<td>C</td>
<td>specifies the name of the OLE property</td>
</tr>
<tr>
<td>out-property-value</td>
<td>C</td>
<td>N</td>
</tr>
</tbody>
</table>

**Details**

The `_getProperty` method is used to get the value of a property of an OLE object.

---

**_getReferenceId**

Returns a reference identifier for use with any OLE-supplied method that requires an OLE object as one of its parameters

**Syntax**

CALL NOTIFY (OLE-object, '_getReferenceId', out-refid);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-refid</td>
<td>C</td>
<td>contains the returned reference identifier</td>
</tr>
</tbody>
</table>

**Details**

The `_getReferenceId` method is used to get the OLE object identifier. The value returned is used in subsequent _do or _compute calls where the OLE-supplied method requires an OLE object as one of its parameters. This value should be used for the object parameter.

**Example**

The following example returns the reference identifier for the OLE object. This identifier is then sent as a parameter value to an OLE-supplied method requiring an object identifier.

```plaintext
   call notify( 'oleobj1',
               '_getReferenceId', refid);
   call notify( 'oleobj2', '_do', 'NewAppl',
               refid, p1, p2);
```
_getType

Returns the object’s type

Syntax

CALL NOTIFY (OLE-object, '_getType', out-type);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-type</td>
<td>C</td>
<td>contains the returned object type</td>
</tr>
</tbody>
</table>

Details

The _getType method is used to get the type of the object. Valid values include Embedded, Linked, Bitmap, Device Independent Bitmap, and Picture.

_inError

Returns an object’s ERROR status

Syntax

CALL NOTIFY (OLE-object, '_inError', error-status<, error-msg>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error-status</td>
<td>N</td>
<td>returns a value indicating whether an automation error has been encountered for the object</td>
</tr>
<tr>
<td>error-msg</td>
<td>C</td>
<td>returns the automation error message</td>
</tr>
</tbody>
</table>

Details

Errors encountered from automation calls can be detected using _inError. The _inError method returns the status of the last automation call and should be called prior to any subsequent automation calls.

Example

The following example detects that an error was encountered during the previous _getPropert call:
call notify('obj1','_getProperty', 'ActiveObject', actobj);
call notify('obj1','_inError', inerror, errmsg);
if inerror then
  link handle_err;

_setProperty

Assigns a value to a property of an OLE object

Syntax
CALL NOTIFY (OLE-object, '_setProperty', in-OLE-property, in-value);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-OLE-property</td>
<td>C</td>
<td>specifies the OLE property name</td>
</tr>
<tr>
<td>in-value</td>
<td>C</td>
<td>N</td>
</tr>
</tbody>
</table>

_Details

The _setProperty method assigns a value to a property of an OLE object.

_update

Updates the object based on its current contents or on the contents of a different HSERVICE entry

Inherited from Widget

Syntax
CALL NOTIFY (OLE-object, '_update', <in-hservice>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-hservice</td>
<td>C</td>
<td>specifies the name of the HSERVICE entry to use to update the object</td>
</tr>
</tbody>
</table>

_Details

The _update method re-creates an object and updates its contents based on its current attributes. The in-hservice parameter is used only with OLE objects and is the name of
an HSERVICE catalog entry. When you specify the in-hservice parameter, the object specified by OLE-object is changed to the object stored in the HSERVICE entry referenced by the in-hservice parameter.

If you use the _update method without specifying in-hservice, the object’s contents are updated with the current OLE object source. This is useful for manually updating a linked object.

Example

In this example, the object stored in OBJ1 is replaced by the SASUSER.EXAMPLES.SOUND1.HSERVICE object:

```r
call notify('obj1','_update_','sasuser.examples.sound1.hservice');
```