APPENDIX

1

SCL Methods for Automating OLE Objects

Summary of OLE Class Methods

Table A1.1 on page 493 contains a list of SCL methods you can use with object linking and embedding (OLE) and indicates which of the OLE classes they apply to.

Table A1.1 SCL Methods Valid for OLE and OLE Automation

Method	SAS OLE class	SAS OLE Automation class
COMPUTE	Yes	Yes
_DISABLE_DEFAULT_ACTION_	Yes	No
_DO_	Yes	Yes
_ENABLE_DEFAULT_ACTION_	Yes	No
EXECUTE	Yes	No
_GET_EVENT_	Yes	No
_GET_PROPERTY_	Yes	Yes
_GET_REFERENCE_ID_	Yes	Yes
_GET_TYPE_	Yes	No
_IN_ERROR_	Yes	Yes
NEW	No	Yes

Method	SAS OLE class	SAS OLE Automation class
_SET_PROPERTY_	Yes	Yes
UPDATE	Yes	No

Note: The _NEW_ method can be used with any class, but the OLE Automation class overrides this method because of special requirements. △

The remainder of this section contains the reference information for these methods.

COMPUTE

Invokes a method on an OLE automation object and returns a value

Syntax

CALL NOTIFY(*OLE-object-name*,'_COMPUTE_',*in-OLE-method*<,*in-parm*…,*in-parm*>,*out-value*);

CALL SEND(*OLE-object-id*,'_COMPUTE_',*in-OLE-method*<,*in-parm*…,*in-parm*>,*out-value*);

Where...	Is type...	And...
in-OLE-method	C	specifies the OLE method name.
in-parm	C or N	provides a parameter to the OLE method.
out-value	C or N	contains the value returned by the OLE method.

Details

The _COMPUTE_ method invokes a method (with parameters) that is exposed by an OLE automation server. The number of parameters (*in-parm* arguments) needed varies among different objects and methods. Only methods that have a return value should be used with the _COMPUTE_ method. For methods with no return values, use the _DO_ method.

Examples

The following example stores the contents of the item in position 2 of an OLE control in the 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 object to 100:

```
call send(oleobj, '_COMPUTE_', 'Cells',
          2, 5, cellobj1);
call send(cellobj1, '_SET_PROPERTY_',
          'Value', 100);
```

_DISABLE_DEFAULT_ACTION_

Disables the OLE object's default action

Syntax

CALL NOTIFY(*OLE-object-name*,'_DISABLE_DEFAULT_ACTION_');

Details

This method prevents the default verb for an OLE object from executing when the object is double-clicked. By default, the default action is enabled.

_DO_

Invokes a method on an OLE automation object with no return value

Syntax

CALL NOTIFY(*OLE-object-name*,'_DO_',*in-OLE-method*<,*in-parm*…,*in-parm*>);

CALL SEND(*OLE-object-id*,'_DO_',*in-OLE-method*<, *in-parm*…,*in-parm*>);

Where...	Is type...	And...
in-OLE-method	C	specifies the OLE method name.
in-parm	C or N	provides a parameter to the OLE method.

Details

The _DO_ method invokes a method (with parameters) that is exposed by an OLE automation server. The number of parameters (*in-parm* arguments) needed varies among different OLE objects and methods. Only methods that have no return value should be used with the _DO_ method. For 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:

```
call notify('oleobj', '_DO_', 'AboutBox');
```

_ENABLE_DEFAULT_ACTION_

Enables the OLE object's default action

Syntax

CALL NOTIFY(*OLE-object-name*,'_ENABLE_DEFAULT_ACTION_');

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-name*,'_EXECUTE_',*in-verb*<,*in-verb*…*in-verb*>);

Where...	Is type...	And...
in-verb	C	specifies the OLE verb to execute.

Details

A list of verbs supported by this object are listed in the Associated Verbs window for the OLE object (after the object has been created). You can specify more than one OLE verb at a time.

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.

_GET_EVENT_

Returns the name of the last OLE control event received

Syntax

CALL NOTIFY(*OLE-object-name*,'_GET_EVENT_',*out-event*);

Where...	Is type...	And...
out-event	C	contains the returned name of the last OLE control event received.

Details

This method returns the name of the event that the specified OLE control most recently sent.

_GET_PROPERTY_

Returns the value of a property of an automation object

Syntax

CALL NOTIFY(*OLE-object-name*,'_GET_PROPERTY_',*in-OLE-property*,*out-property-value*);

CALL SEND(*OLE-object-id*,'_GET_PROPERTY_',*in-OLE-property*,*out-property-value*);

Where...	Is type...	And...
in-OLE-property	C	specifies the name of the OLE property.
out-property-value	C or N	contains the returned value of the property.

Details

The _GET_PROPERTY_ method is used to get the value of a property of an automation object.

_GET_REFERENCE_ID_

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

Syntax

CALL NOTIFY(*OLE-object-name*,'_GET_REFERENCE_ID_',*out-refid*);

CALL SEND(*OLE-object-id*,'_GET_REFERENCE_ID_',*out-refid*);

Where...	Is type...	And...
out-refid	C	contains the returned reference identifier.

Details

The _GET_REFERENCE_ID_ method is used to get the automation object identifier. The value returned is used in subsequent _DO_ or _COMPUTE_ calls where the object method requires an automation 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 automation object. This identifier is then sent as a parameter value to an automation method requiring an object identifier.

```
call notify('oleobj1', '_GET_REFERENCE_ID_',
            refid);
call notify('oleobj2', '_DO_', 'NewAppl',
            refid, p1, p2);
```

_GET_TYPE_

Returns the object's type

Syntax

CALL NOTIFY(*OLE-object-name*,'_GET_TYPE_',*out-type*);

Where...	Is type...	And...
out-type	C	contains the returned object type.

Details

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

_IN_ERROR_

Returns an object's ERROR status

Syntax

CALL NOTIFY(*OLE-object-name*,'_IN_ERROR_',*error-status*<,*error-msg*>);

CALL SEND(*OLE-object-id*,'_IN_ERROR_',*error-status*<,*error-msg*>);

Where...	Is type...	And...
error-status	N	returns a value indicating whether an automation error has been encountered for the object.
error-msg	C	returns the automation error message.

Details

Errors encountered from automation calls can be detected using _IN_ERROR_. The _IN_ERROR_ 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 _GET_PROPERTY_ call:

```
length errmsg $ 200;
call send(objid,'_GET_PROPERTY_',
          'ActiveObject', actobj);
call send(objid,'_IN_ERROR_',inerror, errmsg);
if inerror then
   link handle_err;
```

NEW

Creates a new instance of an OLE automation server

Syntax

CALL SEND(*OLE-instance*,'_NEW_',*new-OLE-id*,*init-arg*,*OLE-auto-app*);

Details

Before you can use SCL code to refer to an OLE Automation server, you must first create an instance of the OLE Automation class.

For more information about the _INIT_ method, see the description of the Object class in *SAS/AF Software: FRAME Class Dictionary*.

Example

The following example creates a new instance of an OLE Automation server and assigns the SCL identifier **exclauto** to the new object. Note that in this example, **Excel.Application.8** is the identifier for Microsoft Excel in the system registry:

```
hostcl=loadclass('sashelp.fsp.hauto');
call send (hostcl, '_NEW_', exclauto, 0,
           'Excel.Application.8');
```

_SET_PROPERTY_

Assigns a value to a property of an automation object

Syntax

CALL NOTIFY(*OLE-object-name*,'_SET_PROPERTY_',*in-OLE-property*,*in-value*);

CALL SEND(*OLE-object-id*,'_SET_PROPERTY_',*in-OLE-property*,*in-value*);

Where...	Is type...	And...
in-OLE-property	C	specifies the OLE property name.
in-value	C or N	contains the value to assign to the OLE property.

Details

The _SET_PROPERTY_ method assigns a value to a property of an automation object.

UPDATE

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

Syntax

CALL NOTIFY(*OLE-object-name*,'_UPDATE_'<,*in-hservice*>);

Where...	Is type...	And...
in-hservice	C	specifies the name of the HSERVICE entry to use to update the object.

Details

The _UPDATE_ method recreates 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 the following example, the object stored in OBJ1 is replaced by the SASUSER.EXAMPLES.SOUND1.HSERVICE object:

```
length refid $ 30;
call notify('obj1','_update_',
            'sasuser.examples.sound1.hservice');
```

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

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–524–8

All rights reserved. Printed in the United States of America.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st printing, September 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.