ABAP/4 OLE Automation Controller

Release 4.6C

SAP ™

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Contents

ABAP OLE Automation Controller

ABAP As OLE Automation Controller

The ABAP programming language supports the OLE2 Automation technique. Desktop applications supporting OLE2 can be thus be called from R/3.

Overview of Topics

For an introduction to this technique, see the example in the corresponding unit of the Tutorial: Communication Interfaces [Ext.].

Introduction

Through its Open Object Interface, ABAP supports the OLE2 Automation technique. Desktop applications that provide their functionality in the form of an OLE2 Automation Server (such as Excel or WinWord) can be thus be integrated into R/3.

All applications controlled by ABAP must be registered in R/3.

The following ABAP key words control the applications:

- CREATE OBJECT
- SET PROPERTY
- GET PROPERTY
- CALL METHOD
- FREE OBJECT

When called from an ABAP program, the SAPGUI acts as OLE client, and the desktop application as the OLE server.
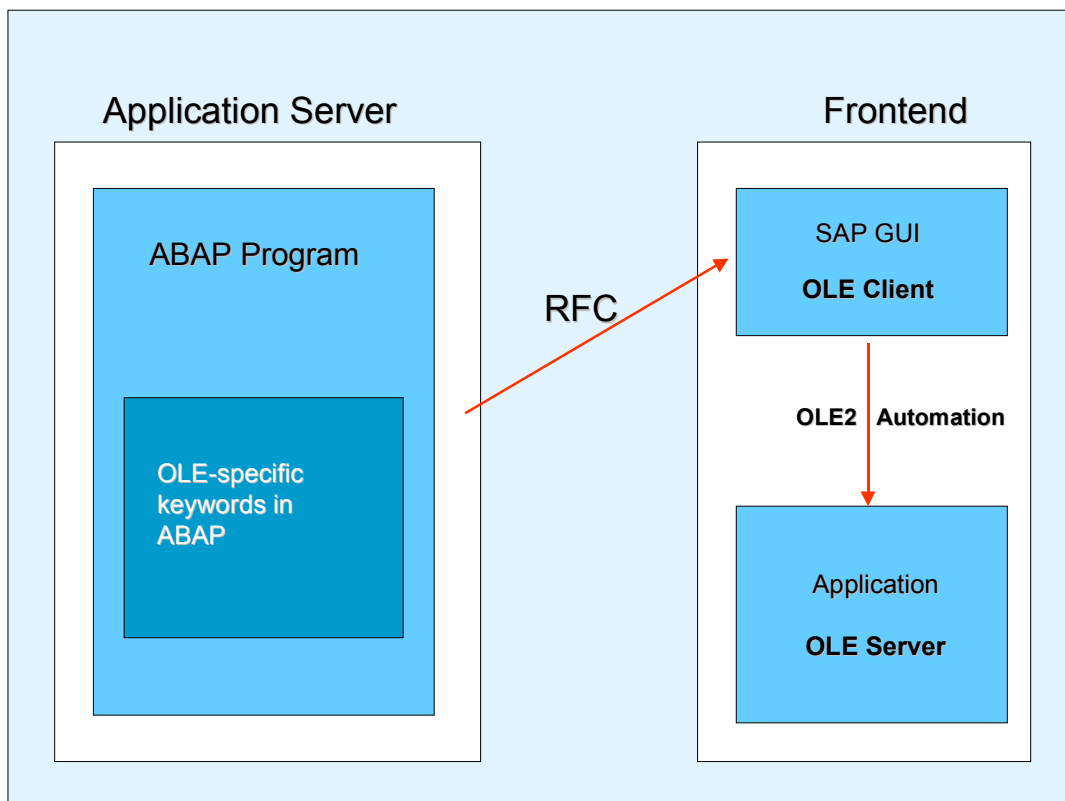
Registering External Applications in R/3

All applications controlled by ABAP must be entered in the table TOLE in the ABAP Development Workbench. In order to maintain table TOLE select *Development → Programming environ. → OLE2 → Configuration* (transaction SOLE).

The key of TOLE (OLE application) is used as object class name in the CREATE-statement. Each entry contains all information needed to generate an OLE object.

Table TOLE also indicates whether type information (Typeinfo) exists for an application.

The **Typeinfo** describes all the objects that a particular application can handle, including all its methods, properties and parameters.

The view "Maintenance view for OLE applications: Overview" displays information on the OLE application names, the version numbers of the OLE applications and class-IDs in character format.

If you want to see more specific information, select *Details* and the view "Maintenance view for OLE applications: Details" with the following fields is displayed:

- OLE application

  The application name determines the object class when an object is generated.

- Version number

  If several versions of an OLE application exist, use the version number from the registration database on the presentation server to distinguish between the various versions.

- CLSID

  OLE class identification in character format as specified in the registration database on the presentation host. The CLSID number is sent to the frontend.

- CLSID LibType

  OLE class-ID in character format. The Type Library of an OLE application partly has its own CLSID.

- OLE object name

  The name of the first OLE object that is created. This field is used in the Object Browser only.

- Typeinfo key

  Key for Typeinfo of an OLE application. Under this key, the typeinfo is stored in the database. NO_TYPELIB indicates that you do not want to use typeinfo for this application.

- Include program

  Include program with constants definitions. Include program has to be maintained manually. This entry is currently used for documentation and reference purposes only.

- Language

  The language of the following text.

- Text

  Description of an OLE application.

Loading OLE Type Information into R/3 For Conversion

You can load the type information from the presentation server into table OLELOAD of the R/3 System (in the ABAP Development Workbench). To do this select *Development → Programming environ. → OLE2 → Load Typeinfo* (transaction SOLI). The view "TypeInfo Loaded" is displayed with the following information:

- Application

- Version

- CLSID of application

- CLSID of object library

With "Load Typeinfo" a part of the typelib is loaded into SAP's database.

This is only possible if the application in question is loaded on your PC.

The ABAP processor can thus perform the necessary type conversions and is not dependent on the language of the OLE application, i.e. an ABAP program can use methods and property names in any language for which type information has been loaded, and can still process applications that have been installed in another language.

For all OLE applications used by SAP standard applications, the English version is delivered in the table OLELOAD.

If an OLE server has no Typeinfo, it can still be processed by ABAP, but the names of the methods and properties have to be in the same language as the language in which the frontend-application is installed.

Conversion Rules

When passing parameters, conversion is always via character type:

The ABAP-type is converted into CHAR according to ABAP-conventions. Afterwards, the CHAR is converted into OLE by using the corresponding OLE types:

ABAP-Type $\rightarrow$ CHAR $\rightarrow$ OLE type

Without Typeinfo, or if the OLE type is undefined, it is set according to the ABAP type as follows:

ABAP types	OLE types
I	Int
P,F	Double
D	Date
T	Time
otherwise	Character

At present, the maximum length of character variables is restricted to 255.

> With some objects the semantics of the parameters depends on the OLE type (see the documentation of the server application).

Using The OLE Object Browser

The Object Browser provides a list of OLE applications with loaded Typeinfo in English. This information is stored in table INDX.

Select *Development → Programming environ. → OLE2 → Object Browser* (transaction SOLO).

You can also create an OLE application interactively within the Object Browser. Then you can invoke its methods or read and write its properties. The results of each call are displayed directly.

> This information has to be loaded by the customer. There are no initial entries delivered by SAP.

Implementation

Communication between the applications server and the frontend is via the RFC interface. The SAPGUI contains special OLE functions which you can address from ABAP programs with normal RFC calls (destination 'SAPGUI').

The implementation is realized by means of the ABAP keywords CREATE OBJECT and FREE OBJECT (which call the function modules CREATE_OBJECT and FREE_OBJECT) and C functions which are written as RFC Server programs.

In CREATE OBJECT [Page 18], the ABAP processor reads the relevant entry in the table TOLE and uses the found OLE class-ID to call the function OLE_CREATE_OBJECT at the frontend via RFC. Then the functions that are necessary for generating an OLE object are called from the Windows OLE library and an object handle is returned to the ABAP processor. From an OLE point of view, the SAPGUI represents the actual OLE client.

The commands CALL METHOD [Page 23], GET PROPERTY [Page 21] and SET PROPERTY [Page 19] are bundled on the application server and passed to the SAPGUI function OLE_FLUSH_CALL in the form of an internal table.

FREE OBJECT [Page 25] calls the SAPGUI function OLE_RELEASE_OBJECT which, besides the OLE call, also releases all resources retrieved for this object at the frontend.

Reading the Typeinfo is also implemented via an RFC call to the SAPGUI.

Examples

Examples and OLE demonstrations are available in WinWord, Excel and Visio. Select *Development → Programming environ. → OLE2 → Demo* (transaction OLE) in the ABAP Development Workbench.

Three OLE servers can be activated using transaction OLE and numerous methods and properties tested by selecting the appropriate pushbuttons:
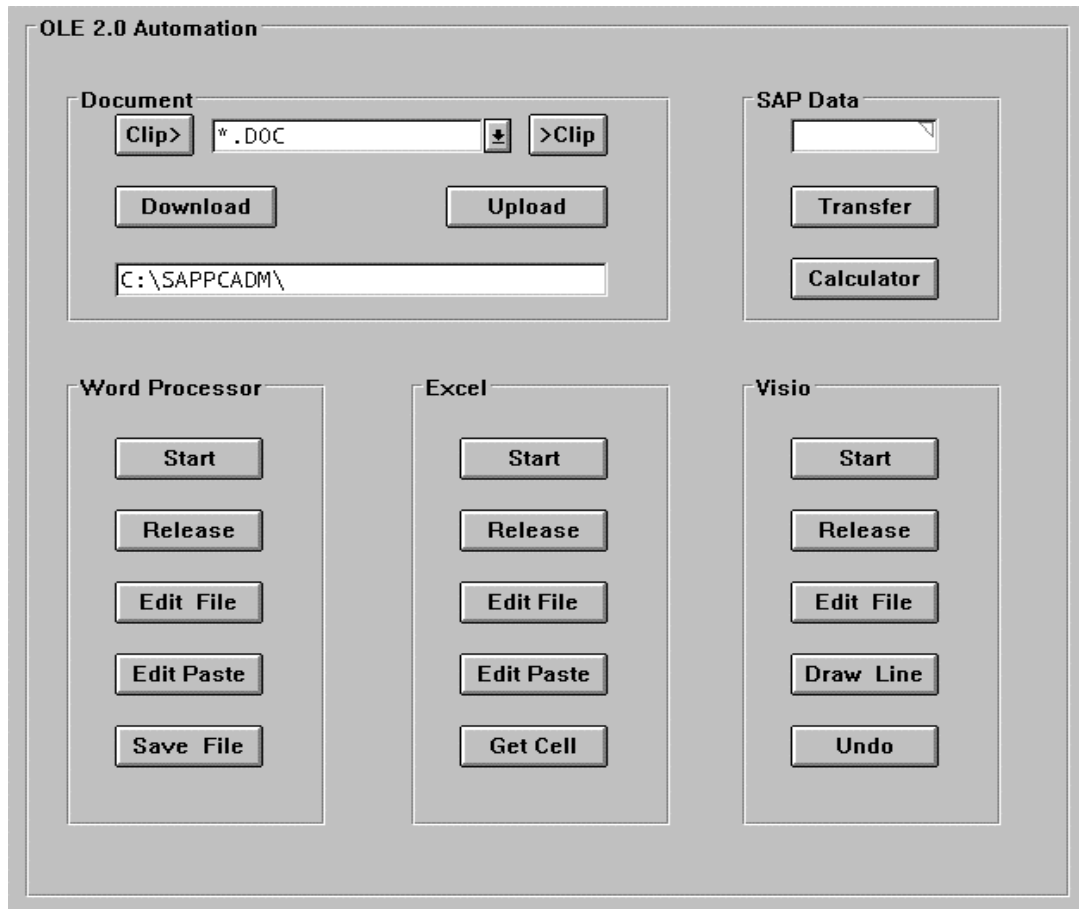
For a simple program example, see the Tutorial: Communication Interfaces [Ext.].

R/3 As OLE Automation Server

In contrast to ABAP as an OLE Automation Controller, R/3 also offers some of its functionality as an Automation Server. This means that all function modules which can be called remotely can be called by any OLE Automation Controller. To do this, the server program RFCSRV.EXE must be installed on the frontend.

For an introduction to this technique, see the example in the corresponding unit of the Tutorial: Communication Interfaces [Ext.].

Related ABAP Keywords

Introduction

From R/3 Release 3.0, the command set of the ABAP interpreter has been enhanced to include key words that allow the application programmer to process external objects. OLE2 was supported as the first object model.

ABAP keywords allow you to control all applications with functionality in the form of an OLE2 Automation Server from an ABAP program:

Examples of such applications are the Microsoft products EXCEL or WinWord.

For further information about OLE2, refer to **OLE2** in the online help (ABAP Editor).

CREATE OBJECT

The ABAP key word CREATE OBJECT generates an object of the class "class".

Basic form: **`CREATE OBJECT obj class.`**

Addition: **`LANGUAGE langu`**

To address an OLE Automation Server (e.g. EXCEL) from ABAP, the server must be registered with SAP. The transaction SOLE (see Registering External Applications in R/3 [Ext.]) allows you to assign an automation server to a class.

The CREATE statement generates the initial object of this class and this can be processed further with the related key words. The return code value of SY-SUBRC indicates the result of the generation. The return code value can be as follows:

- SY-SUBRC = 0:

  Object successfully generated.

- SY-SUBRC = 1:

  SAPGUI communication error.

- SY-SUBRC = 2:

  SAPGUI function call error. The frontend ports of SAP's OLE implementation modules are implemented only under Windows and Apple Macintosh.

- SY-SUBRC = 3:

  The OLE-API call resulted in an error - possibly a storage space problem.

- SY-SUBRC = 4:

  The object is not registered with SAP.

The addition LANGUAGE determines the language chosen for method and attribute names of the object class. If no specification is made, English is the default.

CREATE OBJECT belongs to a group of key words that allows you to process external objects with ABAP. At present, only the object model OLE2 is supported, i.e. all objects must be of type OLE2_OBJECT. This type and other necessary data are defined in the INCLUDE module OLE2INCL.

Generate an EXCEL object.

```
INCLUDE OLE2INCL.
DATA EXCEL TYPE OLE2_OBJECT.
CREATE OBJECT EXCEL 'Excel.Application'.
```

Related topics are SET PROPERTY [Page 19], GET PROPERTY [Page 21], CALL METHOD [Page 23] and FREE OBJECT [Page 25].

SET PROPERTY

The ABAP key word SET PROPERTY sets the property *p* of the object *obj* according to the contents of the field *f.* The object obj must be of type OLE2_OBJECT.

Basic form: `SET PROPERTY OF obj p = f.`

Addition: `NO FLUSH`

Normally, all consecutive OLE statements are buffered by the ABAP processor and sent to the presentation server in bundled form. But it is still possible for a statement to refer to the results of preceding statements.

In debugging, however, you should remember that the values of the return parameters cannot be displayed until directly before execution of the first ABAP non-OLE statement. A command that refers to an object not yet generated by any OLE statement terminates the automatic bundling.

The return code value of SY-SUBRC indicates whether all the bundled commands have been successfully executed. The return code value can be as follows:

- SY-SUBRC = 0:

  All commands were successfully executed.

- SY-SUBRC = 1:

  When communicating with the presentation server, a system error occurred. The field SY-MSGLI contains a short description of the error.

- SY-SUBRC = 2:

  A method call resulted in an error.

- SY-SUBRC = 3:

  Setting a property resulted in an error.

- SY-SUBRC = 4:

  Reading a property resulted in an error.

  In the last 3 cases, a dialog box containing an error note is displayed on the presentation server.

The addition NO FLUSH continues the collection process, even if the next command is not an OLE statement. This means, for example, that a series of properties can be set in a loop and downloaded to the presentation server in a single transport operation.

If NO FLUSH is used, programmers must ensure that they do not rely on the contenst of return parameters that are not yet filled. Also, all objects must be initialized in a bundle, i.e. they must be generated by an OLE call that has already been executed.

Every FREE statement always causes an exchange of the buffer.

SET PROPERTY belongs to a group of key words that allows you to process external objects with ABAP. At present, only the object model OLE2 is supported, i.e. all objects must be of type OLE2_OBJECT. This type and other necessary data are defined in the INCLUDE module OLE2INCL.

SET PROPERTY

Sets the property 'Visible' of an EXCEL worksheet.

```
INCLUDE OLE2INCL.
DATA EXCEL TYPE OLE2_OBJECT.
CREATE OBJECT EXCEL 'Excel.Application'.
SET PROPERTY OF EXCEL 'Visible' = 1.
```

Related topics are:

CREATE OBJECT [Page 18], GET PROPERTY [Page 21], CALL METHOD [Page 23] and FREE OBJECT [Page 25].

GET PROPERTY

The ABAP key word GET PROPERTY copies the property *p* of the object *obj* to the field *f.* The object *obj* must be of type OLE2_OBJECT.

Basic form: `GET PROPERTY OF obj p = f.`

Normally, all consecutive OLE statements are buffered by the ABAP processor and sent to the presentation server in bundled form. But it is still possible for a statement to refer to the results of preceding statements.

In debugging, however, you should remember that the values of the return parameters cannot be displayed until directly before execution of the first ABAP non-OLE statement. A command that refers to an object not yet generated by any OLE statement terminates the automatic bundling.

The return code value of SY-SUBRC indicates whether all the bundled commands have been successfully executed. The return code value can be as follows:

- SY-SUBRC = 0:

  All commands were successfully executed.

- SY-SUBRC = 1:

  When communicating with the presentation server, a system error occurred. The field SY-MSGLI contains a short description of the error.

- SY-SUBRC = 2:

  A method call resulted in an error.

- SY-SUBRC = 3:

  Setting a property resulted in an error.

- SY-SUBRC = 4:

  Reading a property resulted in an error.

  In the last 3 cases, a dialog box containing an error note is displayed on the presentation server.

GET PROPERTY belongs to a group of key words that allows you to process external objects with ABAP. At present, only the object model OLE2 is supported, i.e. all objects must be of type OLE2_OBJECT. This type and other necessary data are defined in the INCLUDE module OLE2INCL.

The addition NO FLUSH continues the collection process, even if the next command is not an OLE statement. This means, for example, that a series of properties can be set in a loop and downloaded to the presentation server in a single transport operation.

If NO FLUSH is used, programmers must ensure that they do not rely on the contents of return parameters that are not yet filled. Also, all objects must be initialized in a bundle, i.e. they must be generated by an OLE call that has already been executed.

Every FREE statement always causes an exchange of the buffer.

Read the property 'Visible' of an EXCEL worksheet:

GET PROPERTY

```
        INCLUDE OLE2INCL.
        DATA:  EXCEL    TYPE OLE2_OBJECT.
               VISIBLE  TYPE I.
        CREATE OBJECT EXCEL 'Excel.Application'.
        GET PROPERTY OF EXCEL 'Visible' = VISIBLE.
```

Related topics are:

SET PROPERTY [Page 19], CREATE OBJECT [Page 18], CALL METHOD [Page 23] and FREE OBJECT [Page 25].

CALL METHOD

The ABAP key word CALL METHOD calls the method *m* of the object *obj*. *m* can be a literal or a variable.

Basic form: `CALL METHOD OF obj m.`

Additions:

1. `... = f`

2. `... EXPORTING p1 = f1... pn = fn`

3. `NO FLUSH.`

Normally, all consecutive OLE statements are buffered by the ABAP processor and sent to the presentation server in bundled form. This means that it is possible for a statement to refer to the results of preceding statements.

In debugging, however, you should remember that the values of the return parameters cannot be displayed until directly before execution of the first ABAP statement external to OLE. Even a command that refers to an object not yet generated by any OLE statement terminates the bundling.

The return code value of SY-SUBRC indicates whether all the bundled commands have been successfully executed. The return code value can be as follows:

- SY-SUBRC = 0:

  All commands were successfully executed.

- SY-SUBRC = 1:

  When communicating with the presentation server, a system error occurred. The field SY-MSGLI contains a short description of the error.

- SY-SUBRC = 2:

  A method call resulted in an error.

- SY-SUBRC = 3:

  Setting a property resulted in an error.

- SY-SUBRC = 4:

  Reading a property resulted in an error.

  In the last 3 cases, a dialog box containing an error note is displayed on the presentation server.

CALL METHOD belongs to a group of key words that allows you to process external objects with ABAP. At present, only the object model OLE2 is supported, i.e. all objects must be of type OLE2_OBJECT. This type and other necessary data are defined in the INCLUDE module OLE2INCL.

Additions

- **Addition 1   `... = f`**

CALL METHOD

Stores the return value of the method in the variable f. The return value can also be of type OLE2_OBJECT. This addition must always come before other additions.

- **Addition 2**   `... EXPORTING p1 = f1... pn = fn`

EXPORTING passes values of fields to the parameters of the method. p1, p2,... are either key word parameters or position parameters.

If assignment of parameters is by sequence, p1, p2,... must begin with "#", followed by the position number of the parameter. At present, only position parameters are supported.

The exporting parameters always come at the end of the statement.

- **Addition 3**   `... NO FLUSH.`

The addition NO FLUSH continues the collection process, even if the next command is not an OLE statement. This means, for example, that a series of properties can be set in a loop and downloaded to the presentation server in a single transport operation.

If NO FLUSH is used, programmers must ensure that they do not rely on the contenst of return parameters that are not yet filled.

Also, all objects must be initialized in a bundle, i.e. they must be generated by an OLE call that has already been executed.

Every FREE statement always causes a download of the buffer.

Open an EXCEL file with the method 'Open':

```
INCLUDE OLE2INCL.
DATA  EXCEL          TYPE OLE2_OBJECT.
DATA  WORKBOOK  TYPE OLE2_OBJECT.

CREATE OBJECT EXCEL 'Excel.Application'.
CALL METHOD OF EXCEL 'WORKBOOKS' = WORKBOOK.

CALL METHOD OF WORKBOOK 'Open' EXPORTING #1 = 'C:\EX1.XLS'.
```

Related topics are:

FREE OBJECT

The ABAP key word FREE OBJECT releases the storage space required for the object *obj.* The object can then no longer be processed.

The return value of *SY-SUBRC* specifies whether the desired object was released or not.

- SY-SUBRC = 0:

  Object successfully released.

- SY-SUBRC = 1:

  SAP GUI communication error.

- SY-SUBRC = 2:

  Error when calling function in SAP GUI.

The OLE function modules are only implemented under Windows.

FREE OBJECT belongs to a group of key words that allows you to process external objects with ABAP. At present, only the object model OLE2 is supported, i.e. all objects must be of type OLE2_OBJECT. This type and other necessary data are defined in the INCLUDE module OLE2INCL.

Release an EXCEL object:

```
INCLUDE OLE2INCL.
DATA: EXCEL     TYPE OLE2_OBJECT.
CREATE OBJECT EXCEL 'Excel.Application'.
FREE OBJECT EXCEL.
```

Related topics are: