SWT
SOFTWARE TOOLS

English User Manual

Duevej 23
DK-2680 Solrød Strand
Denmark
Phone: **+45) 33 33 05 56**
Mail: **swtools@swtools.com**
www: **www.swtools.com**

Object Linking and Embedding

22/11/01 / 2022-09-01 008.384

Contents

1. Preface

SW-Tools Object Linking and Embedding provide you with a simple way to integrate the content of other Windows applications in the report or query defined in a SW-Tools TRIO application. The manual will refer to Object Linking and Embedding as the short name OLE.
For example, you may want to use the word processing features of Microsoft Word when defining a customer letter, or to be able to play a video sequence when querying the article information etc.
The interface provided in TRIO is simply the ability to define a field, which is marked as OLE. Working with the layout of a report or the form of a query you may choose to link to an existing document or create an embedded document.
A link to an existing Microsoft Word document is only a reference to the file name. If you have a document stored as c:/Microsoft/word/customer.doc you may link directly to it. If the document is changed by another user, not working in the TRIO environment, the object will be up to date the next time the report/query is executed.
If using OLE as an embedded document, the actual content of the document is stored together with the TRIO application. Changes to the document object is therefore made only by TRIO using the object application, and cannot be changed directly from the real application.
Choosing between the linking and embedding depends on how you want to store the content of the objects.
This manual will guide you through samples in both RAPGEN and IQ to show the simple use of OLE objects in TRIO.

1.1. Release notes

The following enhancements has been made to the user interface in SW-Tools RAPGEN/IQ:

- New field type OLE in the Free field dialog

Please refer to the samples made in this manual to see the interface changes.

2. OLE in RAPGEN

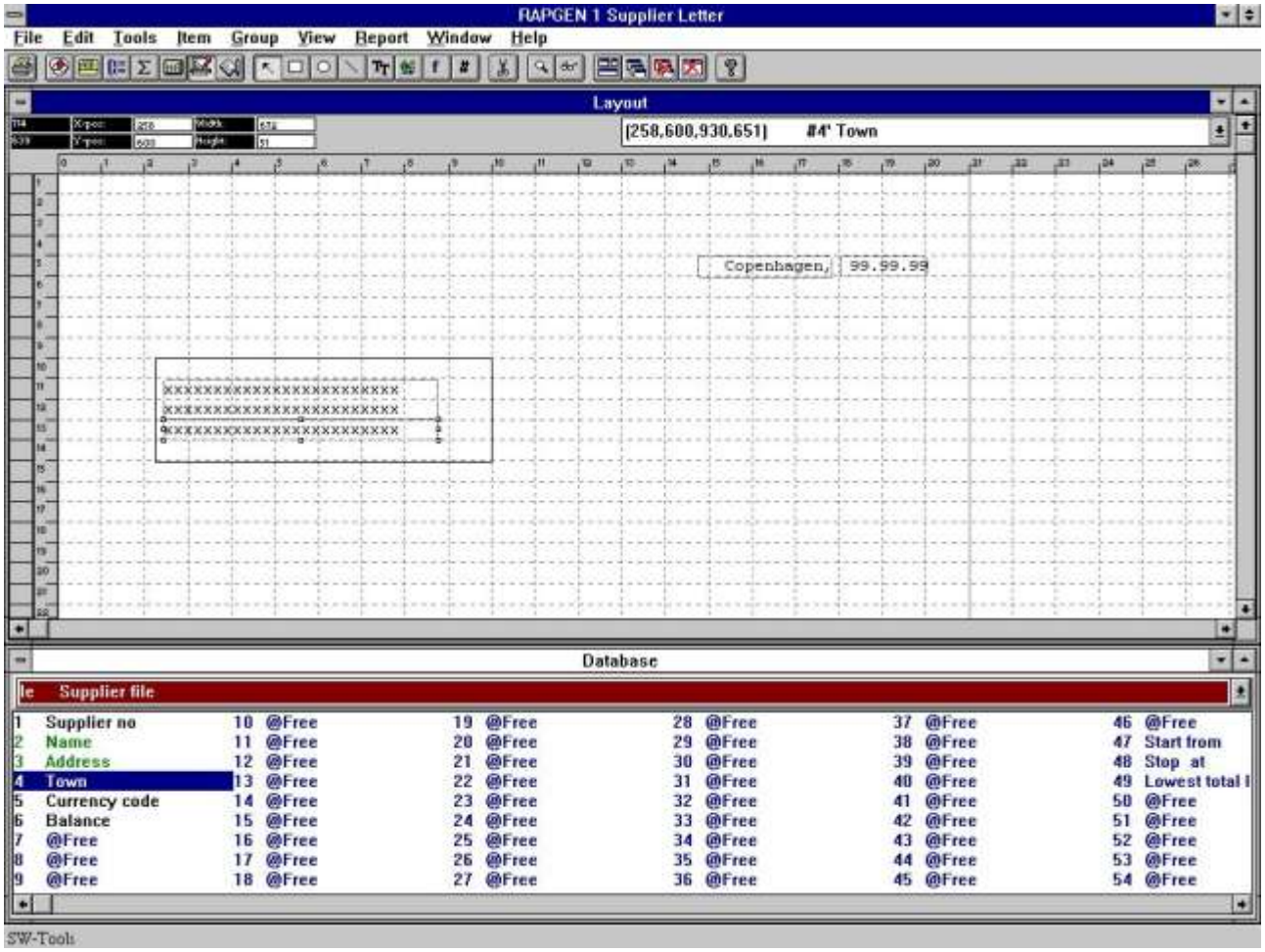
This chapter will step by step describe how you may simply define a letter which will read and print information from the database including text written in Microsoft Word.

2.1. Step 1 - Defining a supplier letter

We start this sample by defining a simple supplier letter, based on the TRIO demo system.

1. Defining the supplier letter

The file 'le' has been selected as main file for the letter and the name changed to 'Supplier Letter'.



2. Adding fields to print on letter

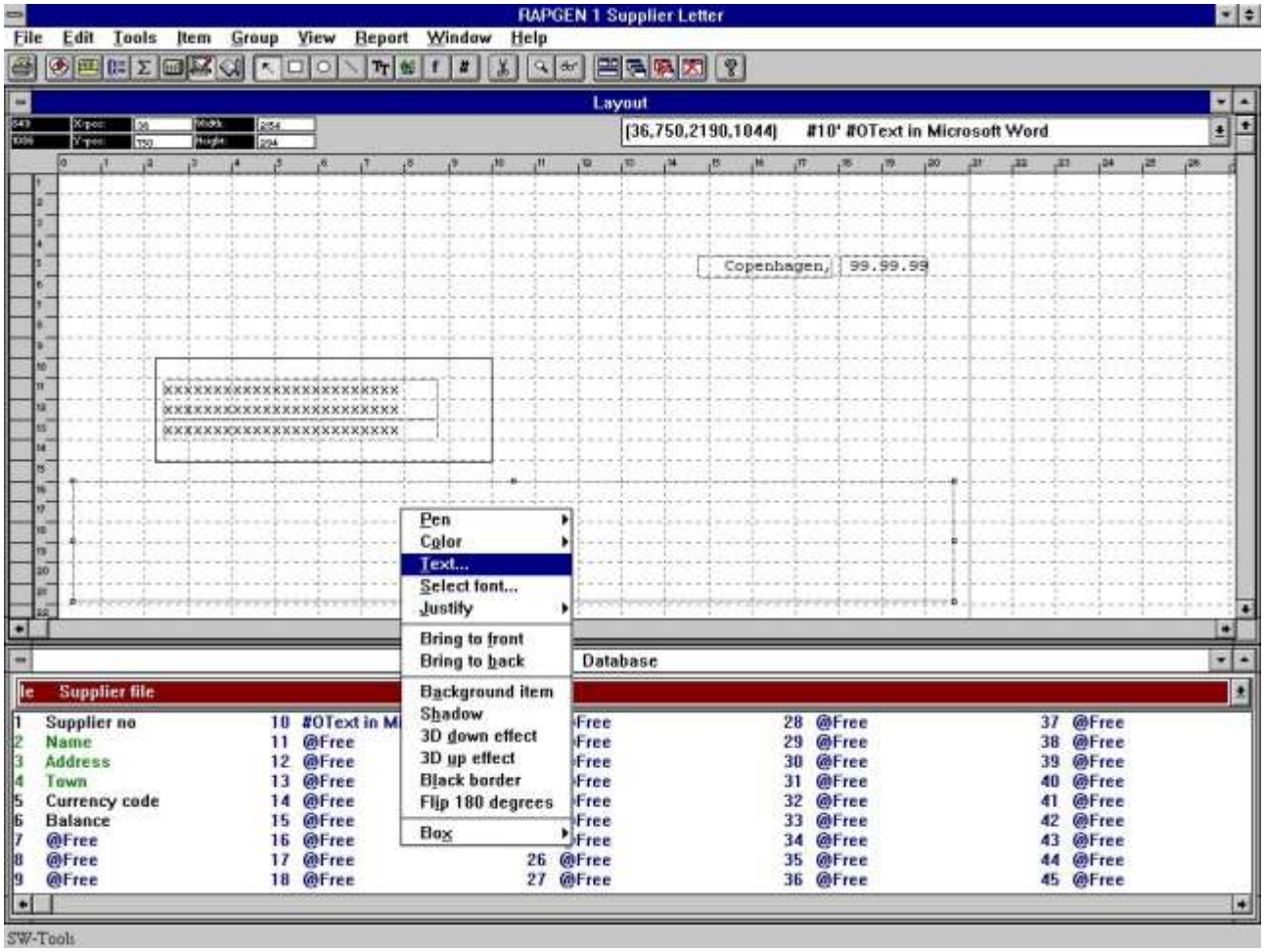
Secondly we insert 3 fields from the main supplier file into the layout.

2.2. Step 2 - Adding the content of a Microsoft Word document as field

To add the content of Microsoft Word to the letter you must first define a field of type OLE 2.0 Object. In this sample we have selected the free field number 10 from the database window.

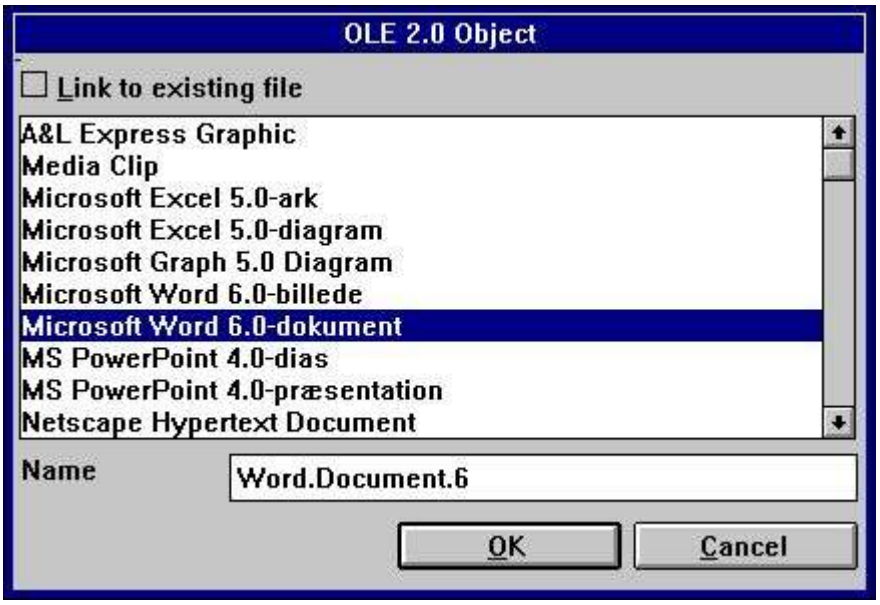
3. Defining the OLE 2.0 Object field

Inserted into the layout and sized accordingly we can select the type of OLE object by clicking with the right mouse button on the field and select the function 'Text...'.



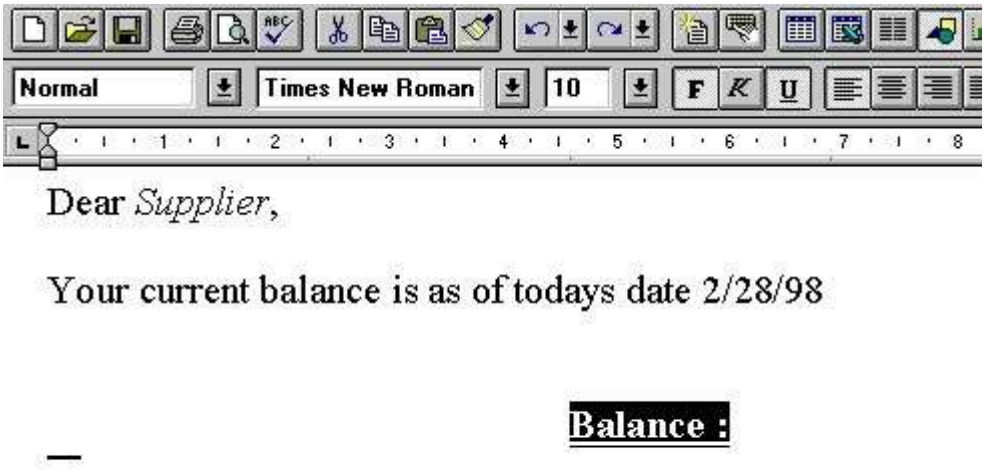
4. How to select the requested OLE object type

The first time you select the 'Text...' function for an OLE object a dialog will appear with all installed OLE 2.0 objects. From this list we select the name



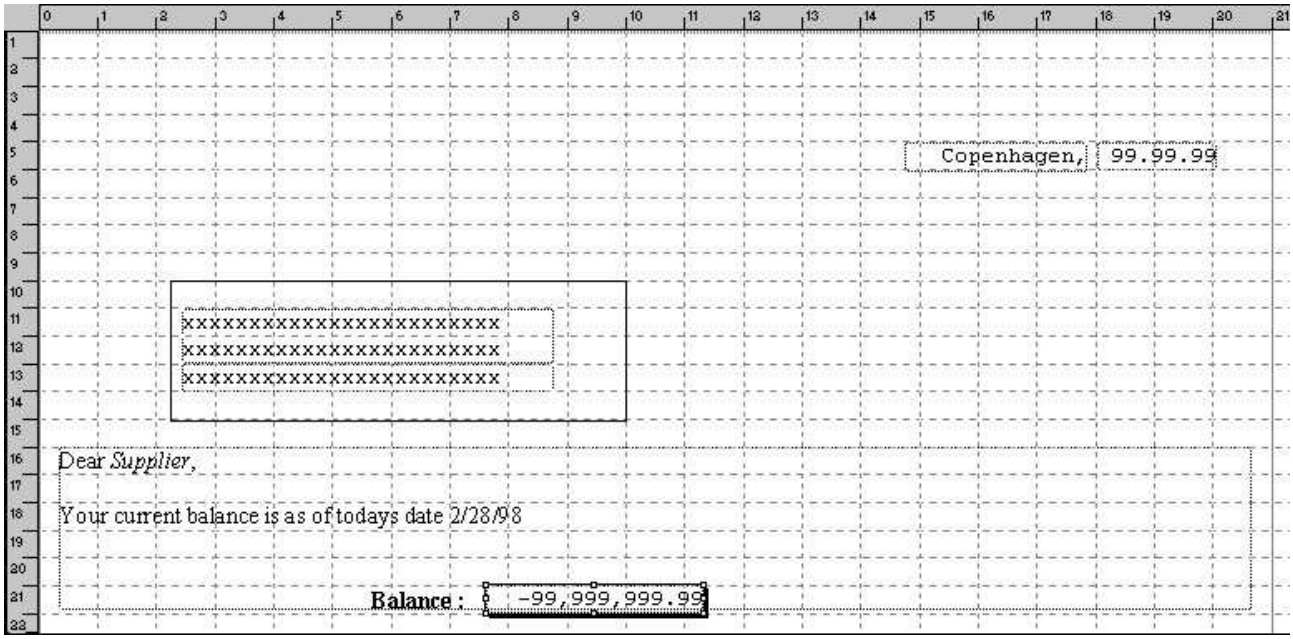
5. Selecting Microsoft Word as object type

Secondly RAPGEN will activate the object server, in this case Microsoft Word, hereby allowing you to use the entire functionality of this application to enter the text in a word processing system.



6. Entering the content of the object in Microsoft Word

When the text has been entered the application can be closed and the object content will appear in the layout of the letter.

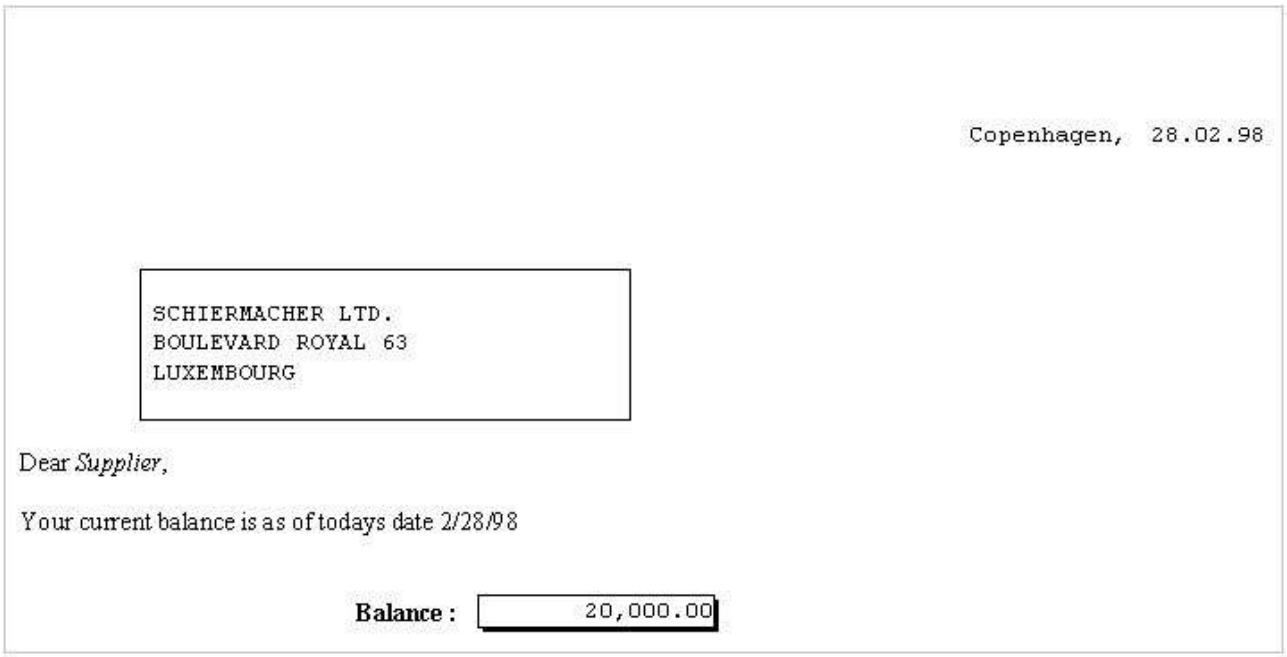


7. Layout of letter including the OLE object content

We finally add the balance field on top of the OLE object field. The letter is complete.

2.3. The final supplier letter

When the letter is printed you will get the following:



8. The printout of the supplier letter

2.4. Using link instead of embedded

The same sample could be done by linking to an existing document file. We have saved the exact same content into a file named

c:/swtools/supplier.doc

and will now link to it instead of having the object embedded in the report.
Insert the field defined as OLE object into the layout and select the 'Text...' function by clicking with the right mouse button on the field.
In the dialog check mark the 'Link to existing file' and enter the file name.

9. Linking to an existing file

The final printout will have the same effect as when embedding the object in the layout.
Please note, you cannot switch from embedded to link directly. You first have to remove the field from the layout and then insert it again. Then you may link to an existing file instead of embedding an object.

3. OLE in IQ/DATAMASTER

This chapter will describe how you may add OLE functionality to any type of IQ/DATAMASTER programs. This section only illustrates the use in IQ because its the same when working with DATAMASTER.
We have chosen a sample which will actually act as a supplier query, where the user may actually add any type of OLE object as extra information on the supplier. This gives you an idea of what may be generated as applications in TRIO when using OLE objects.

3.1. Creating a simple query

This sample is based on a simple supplier query
le#1-6
which has been saved as program 1 in IQ.

10. Simple supplier query in IQ

3.2. Adding the OLE object step by step

Define a **free field #10** in the form function with the name **'OLE Object on supplier'** marked as **'#O OLE 2.0 Object'**.

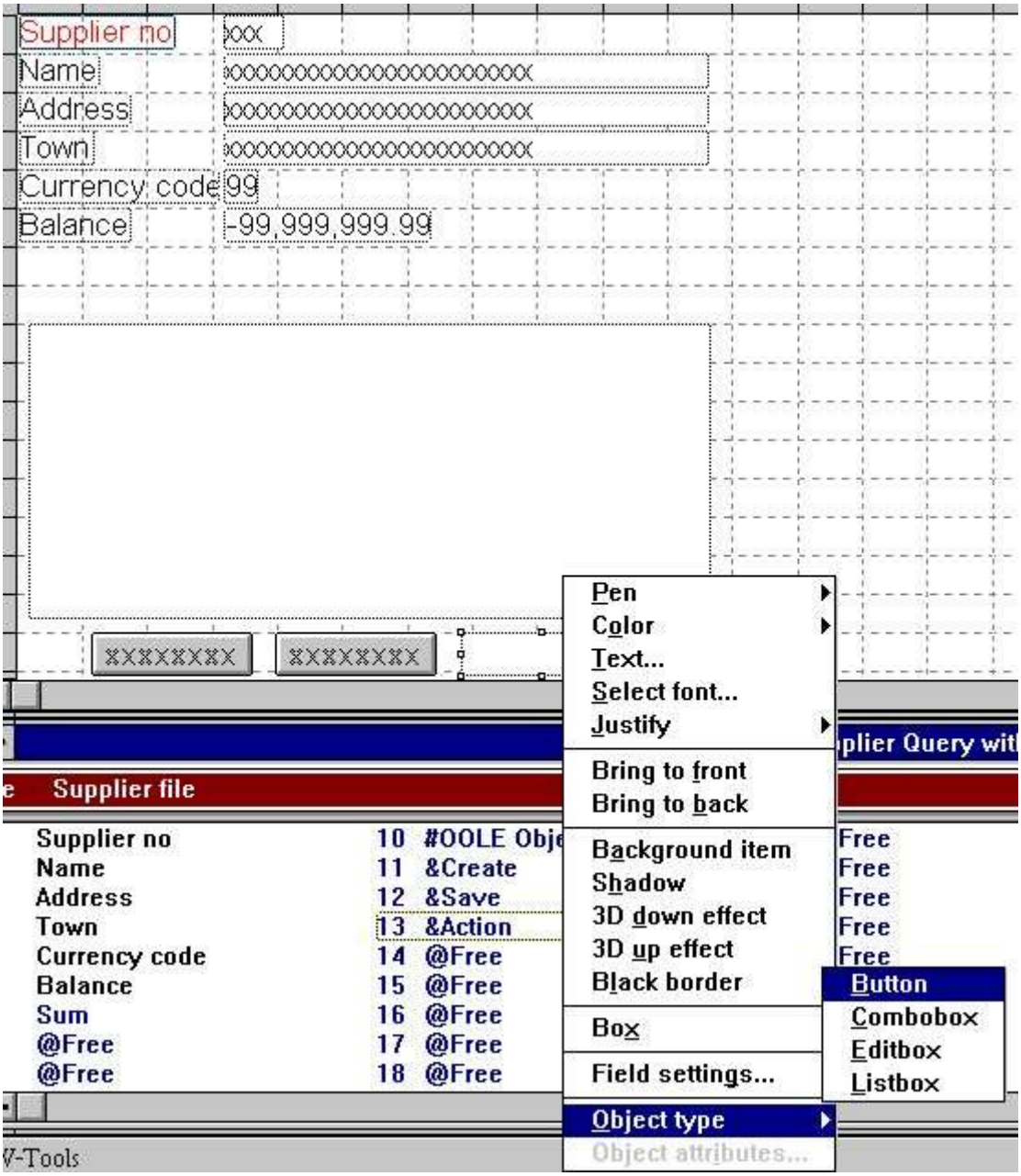
11. Free field defined as OLE object

Insert the free field #10 at the desired position in the form and size the box accordingly. In this sample we have adjusted the box to fit the query window.
After this you add two calculation lines for reading the OLE object **After read of supplierfile**

```
#14="c:/swtools/",#1,".swo"
OleLoad(#10,#14)
```

3.3. Using 3 button fields for OLE functionality

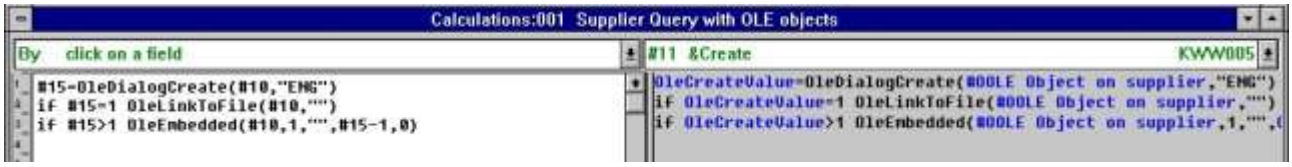
Add 3 free fields defined as **'&Create'**, **'&Save'** and **'&Action'**. All the fields has the format **8**. When inserted in the layout, mark the fields as object type **'Button'** clicking the right mouse button on each of the 3 fields.



12. Buttons to control the OLE object

3.4. How to create the object

The calculations needed to create an object as embedded or linked may simply be by the following calculations:

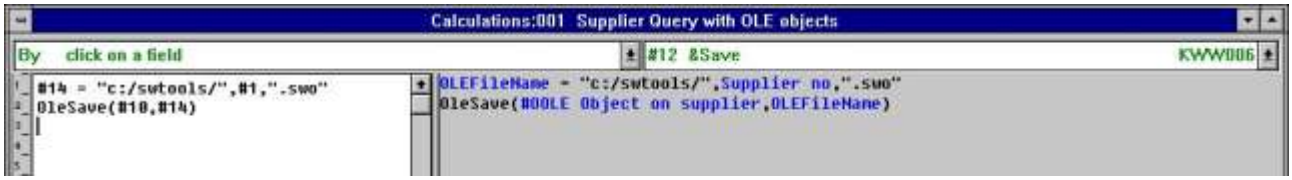


13. Calculations for creating an OLE object

The free field #15 used to retrieve the selected object is defined as format '9,'.

3.5. How the object can be saved

To enable the user to save the created object, we have added the following calculations:



14. Calculations for saving the created OLE object

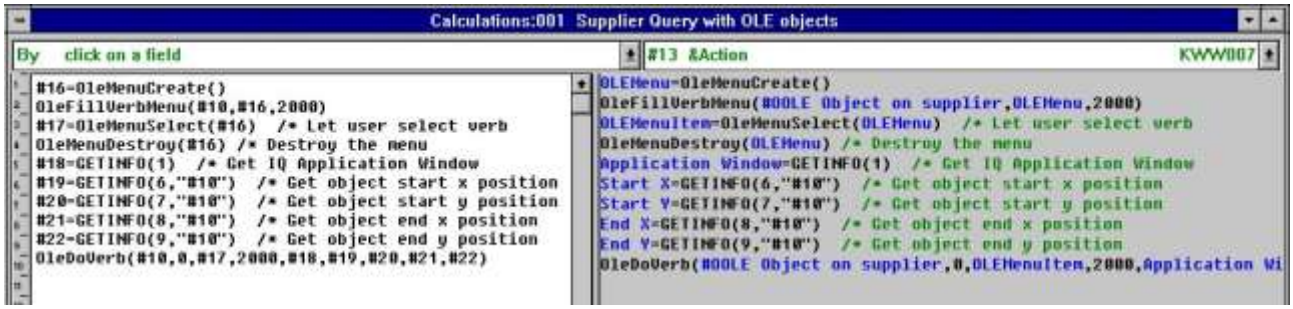
The default path **c:/swtools/** and the **supplier no** plus the extension **.swo** generated the complete object file name. For example, for supplier no 205 the file name will be

c:/swtools/205.swo

The format of the free field 14 is **128** (must be an alpha numeric field to hold a file name including the path).

3.6. Performing actions on the object

Because an OLE object may be of different types, the action a user may perform on the object can only be determined by the object application. For example, it an object is of type Microsoft Word, a user may normally only edit the object, where as for an Microsoft Video object, the user may not only edit the content of the object but also play it. The calculations added are a standard sample of how to let the user select the action to perform on an object of any type.



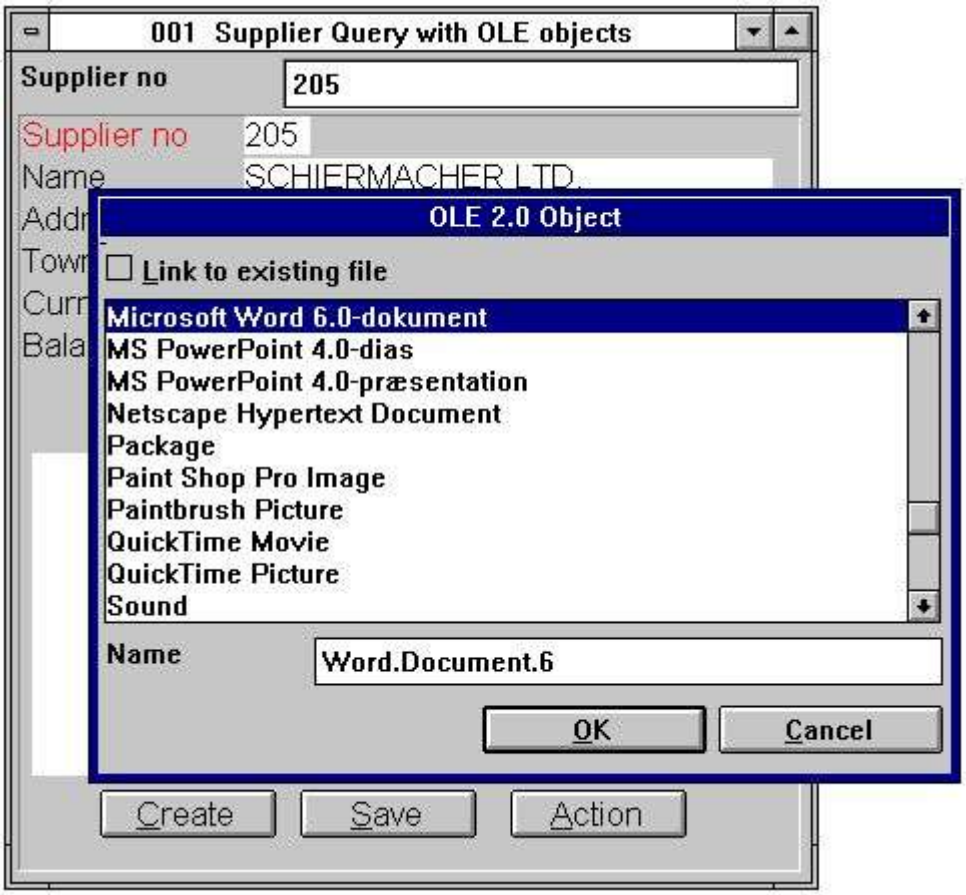
15. Calculations for performing actions on the OLE object

You may refer to the description of the functions used. The free fields used are defined as the following:

Field no	Name	Format
16	OLEMenu	9,
17	OLEMenuItem	9,
18	Application Window	9,
19	Start X	9,
20	Start Y	9,
21	End X	9,
22	End Y	9,

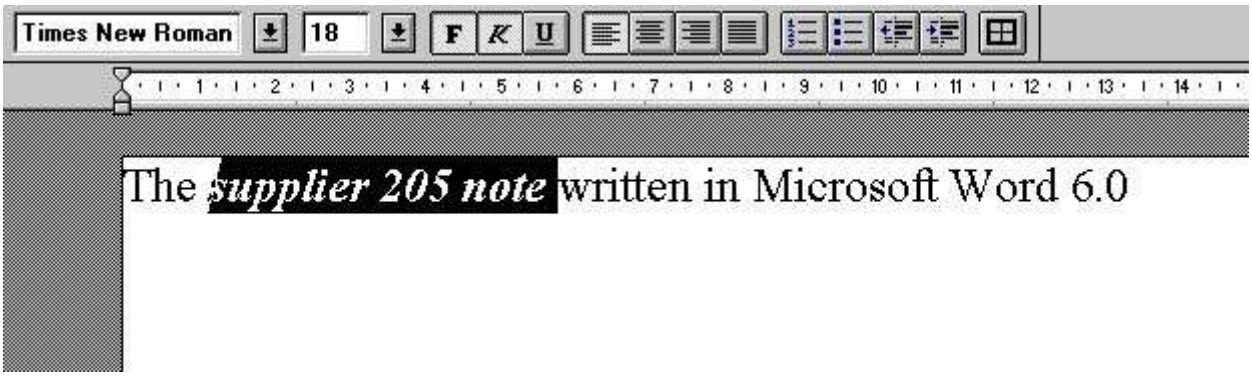
3.7. How to work with the final query

When working with the defined query, it is with the normal functionality of IQ, but extended with the 3 special calculated buttons.
Let say the user locates the supplier no 205, by entering the value 205 in the query key field.
The user now wants to write a note on this supplier using Microsoft Word. The first thing the user may do, is to click on the button **Create** which will present the dialog for creating an object. From this dialog the user selects the 'Microsoft Word' as an embedded object.



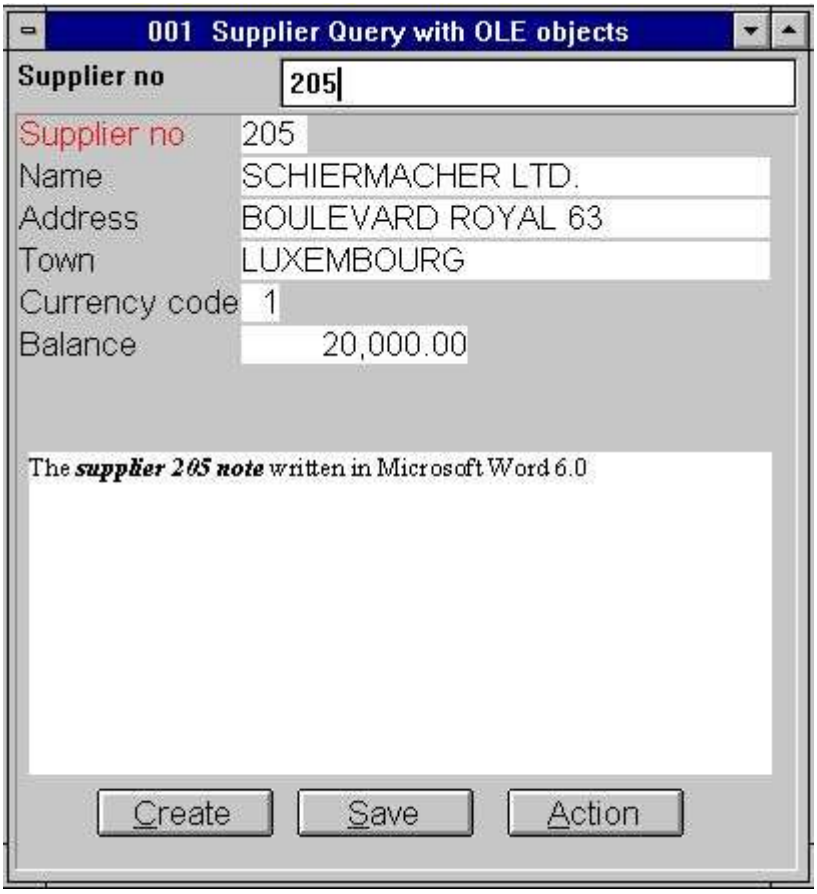
16. User creating an object for supplier no 205

When the object has been created the user may activate the object application Microsoft Word by clicking on the button **'Action'** and select the **Edit** function.



17. User entering the note in Microsoft Word

When the user closes Microsoft Word the content of the object is displayed within the TRIO IQ query.



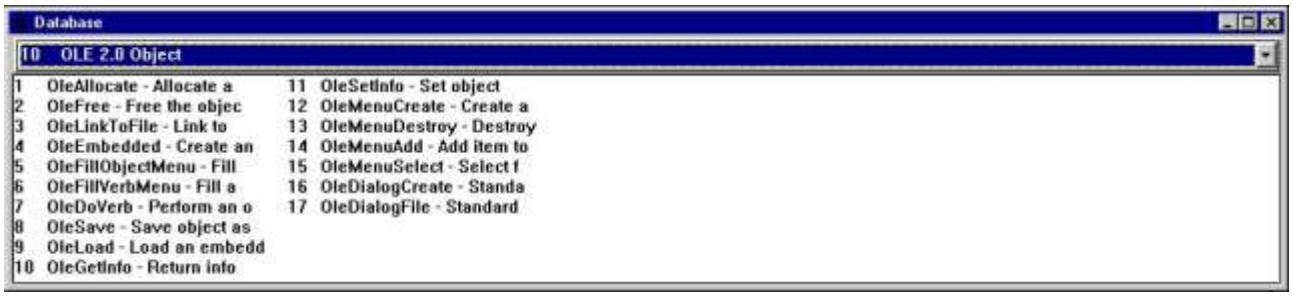
18. IQ Query with user written note in Microsoft Word

4. OLE functions

The collection of OLE functions are installed as a Windows DLL (Dynamic Linked Library). All the functions are described and implemented as sub functions in TRIO in the 10.xxx file, where xxx is the language code, located in the TRIO installation directory.

4.1. View of the on-line documentation

The on-line description of the OLE function may be viewed directly from the database window. Select the file named 'OLE functions', which is file id 10.



19. On-line documentation of the functions

4.2. General error codes

The following list contain all general error codes:

00 No error
01 Missing OLEx.DLL
02 Missing function in DLL
03 Cannot set message queue
04 Subcall of function failed
05 Call of OLE function failed
06 Illegal OLE version
07 Must call ole_init first
08 Windows register function failed
09 Missing COMPOBJ.DLL
10 Missing STORAGE.DLL
11 Object does NOT support Unknown interface
12 Missing interface
13 Cannot allocate needed memory
14 No verbs found for this object
15 Unknown verb or menu id
16 No objects for OLE2.0
17 Unknown object or menu id
18 No storage allocated for object
19 No call has been made to ole_init
20 Missing OLE2DISP.DLL/OLEAUT32.DLL
21 Cannot convert string to OLE string or visa versa

4.3. **OleAllocate** - Allocate a new object

This function allocates a new OLE object. The return value is used as first parameter for almost all other OLE functions.
When the object is no longer needed it should free the memory used by calling the function OleFree().
Please note, a OLE field inserted in the layout of a report or in a form on a query/data entry program is automatically allocated by start and will free memory by termination.

```
OleFree(#50) /* Free memory used by object
```

4.4. **OleFree** - Free the object

This function will free the memory allocated by the OleAllocate() function.
Please note, a OLE field inserted in the layout of a report or in a form on a query/data entry program is automatically allocated by start and will free memory by termination.

See OleAllocate

4.5. **OleLinkToFile** - Link to an object file of any type

This function can make a link to an existing file of any type and hereby load the object. For example, if the filename given in *par2* is "notice.doc" the function will let OLE search the registry for the extension ".doc" and find that this is a document of type 'Microsoft Word'.
The function will clear the current content of the object *par1* before making the link.
With the function OleDialogCreate() you may simplify the creation of linked and embedded objects my means of a standard Windows dialog.
When using a linked object in TRIO, only the filename will be saved when used in layout/form. This also applies for any files saved with OleSave(). If it is required to have a separate object, owned by the actual TRIO application, use an embedded object instead.
Note 1 If *par2* equals "" (no filename) the function will use the current selected file name. For example, if a prior call to OleDialogCreate() has been made and the user has selected the file named "sheet.xls", it is the current selected file name.

See OleAllocate

4.6. **OleEmbedded** - Create an embedded object

This function will create an embedded object, e.g. an object owned by the TRIO application. Before an embedded object can be created it is required to pass the object program id. The program id is known from the Windows registry (Refer to the object documentation).
For example, to create a Microsoft Word 6.0 document, the program id is "Word.Document.6".
The parameters *par4* and *par5* is only used when a prior call has been made to OleDialogCreate() or OleFillObjectMenu()/OleMenuSelect(). Please refer to the documentation on these functions.

OleEmbedded(#50,0,"Word.Document.6",0,0)

4.7. **OleFillObjectMenu** - Fill a menu with all registered objects

This function may be used when creating a tracking menu for the user to select an object. The function will add items to a menu created by OleMenuCreate().
Each item added to the menu will have a unique id, from 0 to x. Because this may conflict with other menu items added prior to this function call, you may use *par2* as a menu offset, e.g. if *par2* is 1000 the id's of the added items are numbered 1000 to x.
The number of items added depends on how many applications has been installed on the system, which may a lot of objects. Due to display limitations of a tracking menu, a maximum number of items may be given as parameter *par3*, e.g. if *par3* equals 10 a sub menu is created for each 10 items. The sub menu will be names xxxxx 1-10, xxxxx 11-20 etc., where the xxxxx must be given in *par4*.

OleFree(#50) /* *Free memory used by object*

4.8. **OleFillVerbMenu** - Fill a menu with the object verbs

This function may be used to fill a menu with all verbs of an object. A verb is the action that may be performed on the object. For example, a verb may be Edit, Open, Play, etc., dependent on the object type. Normally, all objects have the verbs Edit and Open, where some objects have additional verbs like Play (Sound/Video).
The verb is uniquely identified by a number, only known to the object. By using this function it is possible to add items to a menu, where from the user may select the action to perform on the object.
Each item added to the menu will have a unique id, from 0 to x. Because this may conflict with other menu items added prior to this function call, you may use *par3* as a menu offset, e.g. if *par3* is 1000 the id's of the added items are numbered 1000 to x.

OleDoVerb(#50,0,#52,2000,#53,#54,#55,#56,#57)

4.9. **OleDoVerb** - Perform an object verb

This function to select a verb for an object. A verb is the action that may be performed on the object. For example, a verb may be Edit, Open, Play, etc., dependent on the object type. Normally, all objects have the verbs Edit and Open, where some objects have additional verbs like Play (Sound/Video).
If the object verb id is known it may be parsed directly in *par3* with *par2* set to 1. If not, the example from OleFillVerbMenu() may be used (the user may select the action to perform on the object from a menu).
The parameters *par5-par9* is required and may be obtained by the standard TRIO sub function GETINFO(). The parameters must be defined as a field format "8." (16/32 bit compatible). If wrongly defined the application may result in errors or unexpected results.

See OleFillVerbMenu

4.10. **OleSave** - Save object as embedded into file

This functions can save an object into a file. The object to save may be linked or embedded. If it is a link, only the file name is saved, as for an embedded object the entire object is saved. The saved object may be loaded again using the OleLoad() function.

OleLoad(#50,#51) /* *Load the object*

4.11. **OleLoad** - Load an embedded object from file

This function can load an object saved prior with OleSave().

See OleSave

4.12. **OleGetInfo** - Return information about object

The function returns information about the object. The information returned depends on the mode (*par2*), which can be one of the following:
With mode 0 *par3* is not used. With mode 1 it will copy the file name of the object into *par3*

end

4.13. **OleSetInfo** - Set object Information

The function will set information about the object. The information to set depends on *par2*, which can be one of the following:
With mode 0 the object flags must be parsed in *par3*. The value may be one or more of the following values added together:
0 - No flags. The object is scaled according to the field box dimensions. 1 - The object uses its real dimensions. If the field box size is smaller than the object it is clipped. 2 - The vertical dimension is scaled according to the horizontal dimension of the field box. 4 - The horizontal dimension is scaled according to the vertical dimension of the field box.
Any object created uses flag 2 as default.

OleSetInfo(#50,0,0) /* *Scale the object according to the field box*

4.14. **OleMenuCreate** - Create a menu

The function may be used to create a Windows menu. When a menu has been created items may be added using the OleMenuAdd function.
When the menu is no longer needed it must be destroyed using the OleMenuDestroy function. If it is not, memory is not released!

```
if #52=3001 OleLinkToFile(#50,"c:/swtools/document.doc")
```

4.15. OleMenuDestroy - Destroy a menu

This function will free all memory used by a menu created with the OleMenuCreate() function. If any sub menu has been added to the menu parsed in *par1* it will be destroyed also.

See OleMenuAdd

4.16. **OleMenuAdd** - Add item to a menu

This function may be used to add items to a menu created with OleMenuCreate(). The *par2* controls the type of item to be added to the menu:
If *par2* is 0 a text item is added to the menu.
If *par2* is -1 a separator (dividing line) is added to the menu. With this mode *par3* and *par4* is not used.
If *par2* is greater than 0 it must be another unique menu id returned from the OleMenuCreate() function.
The parameter *par3* must be a unique menu item id, which will be returned when the user selects an item from the menu, and *par4* simply contains the text of the item.

OleMenuDestroy(#51) */* *This will free all menus created in #51-53*

4.17. **OleMenuSelect** - Select from menu at the current cursor location

The function activates a floating menu at the current cursor location on screen. The menu will remain active on screen until the user selects one of the items or clicks outside the menu.

See OleMenuAdd

4.18. **OleDialogCreate** - Standard dialog for creating embedded and linked objects

This function may be used to simplify the creation of an object.
The *par2* is the language id, which controls the dialog text used.

if #51>1 OleEmbedded(#50,1,"",#51-1,0) /* *Create embedded using the returned menu item id*

4.19. **OleDialogFile** - Standard dialog for selecting file name

This function will let the user select a file name using the standard Windows file dialog.
The window handle in *par1* may be obtained using the subfunction GETINFO().
The dialog requires two parameters to control the file name filters. The parameter *par2* is used for the extension of a file name, e.g. "ole" as extension will only list all files named "xxxxxxxx.ole". The parameter *par3* is the definition of the filter including a filter description. For example, a filter may be defined as
"All files,*.*"
which defines a filter for the user to select with the description "All files" and the extension "*.*". It is important that the description and extension is separated by a comma, otherwise the dialog may fail to be displayed!
If the mode (*par4*) is 1 the file dialog will let the user enter a new file name or select an existing file name for save. If an existing file name is selected the user must answer yes in order to overwrite the file.
If a file name is parsed in *par5* and no value parsed in *par6* ("" empty text) the dialog will use the path from the file name in *par5*. For example, if *par5* is "c:/swtools/sheet.xls" the initial path will be "c:/swtools".

See OleGetInfo()

5. Technical specifications

5.1. Requirements

The OLE interface requires SW-Tools TRIO version 007.001 or higher. It will be supported in 16 and 32-bit versions.

5.2. Files installed

10.eng	Subfunction descriptions and interface for SW-Tools TRIO calculations
ole-eng.hlp	Window on-line manual
swo999xx.dll	OLE Dynamic Link Library, where 999 is the major version number and xx is 16/32 bit

Figure list

Index