

enaio[®]

Software Documentation
enaio[®] editor-for-events

Version 8.50

All software products as well as all related extension programs and additional functions are registered and/or in-use trademarks of OPTIMAL SYSTEMS GmbH, Berlin or its subsidiaries. They may only be used according to a valid licensing agreement. The software as well as related documentation are protected by German and international copyright law. Unauthorized duplication and sales is plagiarism and subject to criminal prosecution. All rights reserved, including reproduction, transmission, translation, and storage with/on all kinds of media. For all preconfigured test scenarios or demo presentations: All company and person names which occur in examples (screenshots) are fictional. Any resemblance to existing companies or persons is purely coincidental and unintentional.

Copyright 1992 – 2018 by OPTIMAL SYSTEMS GmbH
Cicerostraße 26
D-10709 Berlin

28.02.2018
Version 8.50

Contents

Contents	3
Introduction.....	4
About the Manual.....	4
About enaio® editor-for-events	4
Events for enaio® webclient.....	5
Installation, Licensing, Security System	5
Events	6
Quick Introduction	6
Client-Side Events.....	8
Events for Changes in Batch Mode	14
Event 'StartAction'	15
Handoff Files.....	15
Structure of Handoff Files	17
Client-Side Handoff Files.....	20
Handoff Data of Tables.....	44
The ActiveX Control OXACTIVE.DLL.....	45
Methods.....	45
Objects	46
Examples of Use in VBScript.....	51
Server-Side Events	52
Server Events for the Archiving Process.....	53
Server-Side Scripts.....	54
Script Development	54
RunScript.....	61
Global Scripts.....	62
Controlling the Info Window	64
Event Administration	65
enaio® editor-for-events.....	67
enaio® editor-for-events – Introduction.....	67
Creating Events.....	67
Importing Scripts.....	69
The Editor Window.....	70
Export/Import.....	72
Debugging.....	73
Index.....	76

Introduction

About the Manual

The manual is available as a PDF file which is located in the documentation directory.

All procedures described in the manual are based on mouse operation and use of the ribbon buttons. However, all operations can be performed with the keyboard as well. enaio® editor-for-events follows the conventions of MS Windows. Use **Alt** plus the underlined letters in the menu.

About enaio® editor-for-events

An event is a VB script which is assigned to an action in enaio® client with a DMS object or an application situation, and is automatically started from enaio® client by the user action. A VB script can, for example, carry out a validity check or automatically complete data when saving the indexing of a DMS object.

Events can also be assigned to server jobs. Thus, a VB script can be run before or after a server job is executed.

Create events using enaio® editor-for-events. As a component of the enaio® content management, workflow, and archive system, it is an integral part of enaio® client. Given that a user is provided with all necessary system roles and licenses, the corresponding functions are activated in enaio® client.

For most events, enaio® client creates a handoff file with contextual data which can be accessed for viewing and editing by use of a VB script. Some events require a return value which can be entered into the handoff file. Other events update data on the basis of modified entries in the handoff file. According to the return value, enaio® client continues to execute the action or stops its execution.

The integrated ActiveX control `oxactive.dll` offers methods and objects that allow reading and editing data of the handoff file in a script.

What is more, VB scripts may be used to execute actions independently of enaio® client or to start the execution of actions in enaio® client independently of the handoff file via the COM interface. Details on the COM interface can be found in the 'enaio® client programming reference' documentation.

Before being saved to the database, events are encrypted but may also be saved and exchanged as files.

The events written by OPTIMAL SYSTEMS will be provided as files in encrypted format which can be imported and assigned to DMS objects with enaio® editor-for-events.

To assign events to users, enaio® administrator must be used.

Events for enaio® webclient

Events for enaio® webclient are also created with enaio® editor-for-events, but in JavaScript not VBScript. The documentation can be found online under:

https://help.enaio.com/blue/03_dev-doc/webclient/wc_con_scripting.htm

The enabling and disabling of events in enaio® administrator (cf. 'Event Administration') applies only for the enaio® client. Events in enaio® webclient are always executed.

Installation, Licensing, Security System

The components of enaio® editor-for-events will be automatically installed during installation of enaio® client.

To enable the creation of events, the workstation must be provided with the 'EVE' license, the 'ASC' client license, and the system role 'Client: Create events'. Additionally, the system role 'Client: Debug events' is required if events are designated to be tested.

To import events that were created by OPTIMAL SYSTEMS, the 'ASC' client license and the system role 'Editor: Customize database' are needed.

To delete events, you require the system roles 'Editor: Customize database' and 'Clients: Create events'. In both cases, the 'EVE' license is not required.

The 'SDE' license is necessary if the 'dtr.SynchronizeData' server job is used as event to transfer data.

In the enaio® client settings dialog, activate the access to the functions (see 'Creating Events') so that they are available in the 'Object search' area.

Events

Quick Introduction

An event is started by enaio® client or enaio® server as a result of a user action, an application situation, or a job execution, and executes a script.

Licenses are required to be installed and necessary system roles must be respectively assigned in order to enable the utilization of events (see 'Installation, Licensing, Security System' and 'Event Administration').

A distinction must be made between client-side and server-side events:

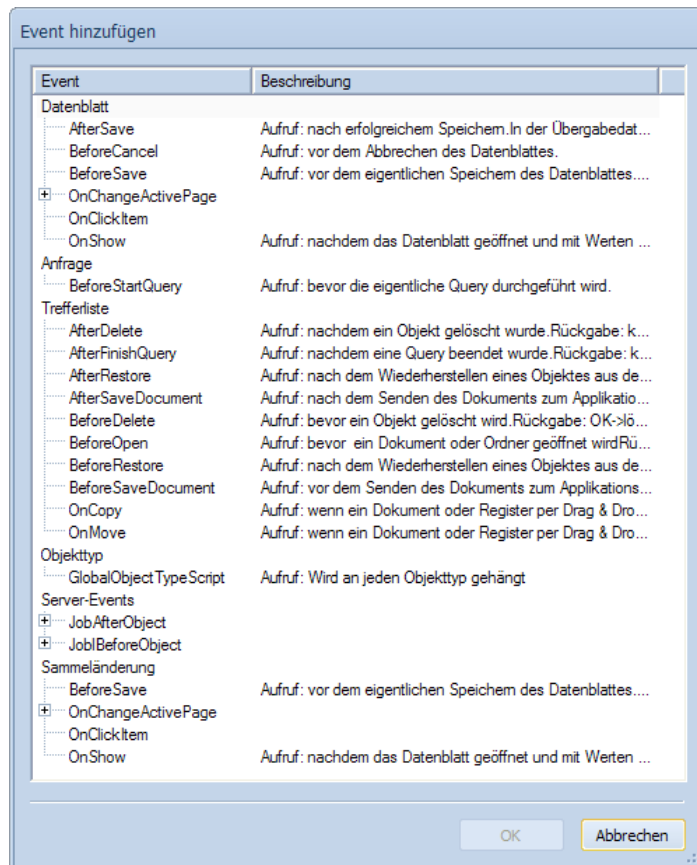
- § Client-side events (see 'Client-Side Events') are triggered by actions in enaio® client, for example when an object is shown.
- § Server-side events, on the other hand, are assigned to server jobs (see 'Server-Side Events').

Events allow, e.g. automatically completing particular fields when saving the index form in case these have been left empty by the user: if the editor did not insert his name into the 'Editor' field, an event can automatically fill in the name before the index form is saved. You can use the event **BeforeValidate** for this.

In order to create such a client event, open the context menu of the DMS object in the 'Object search' area and select the **Add event item**:

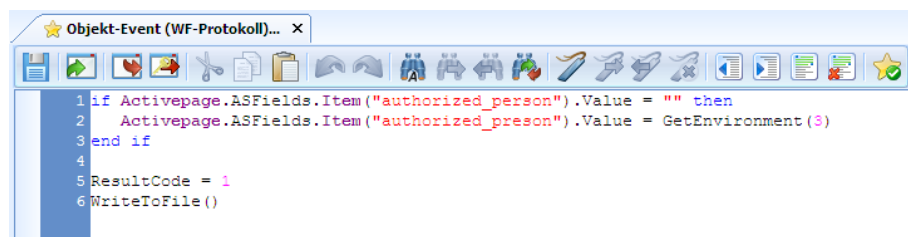


Select the **BeforeValidate** item from the **Add event** dialog and confirm with **OK**:



As a result, enaio® editor-for-events will open (see 'For searches in enaio® client, users can configure the query behavior in the 'Query behavior' area of their user-specific personal settings dialog.

enaio® editor-for-events'). In this editor you enter the VB script code to be executed by the **BeforeValidate** event.



Insert the following script into the editor:


```
if ActivePage.ASFields.Item("Editor").Value = "" then
    ActivePage.ASFields.Item("Bearbeiter").Value = GetEnvironment(3)
end If
ResultCode = 1
WriteToFile()
```

To query whether or not the user has entered his name into the 'Editor' field on the index form (line 1), the **ActivePage** object is used to access the field's value. This object represents a reference to the active data sheet of the index form (see 'Objects').

As the method **Item** of the collection **ASFields** includes all fields of the form, it can be used to access a particular field on the index form (here: the 'Editor' field). To address a particular field, its name must be passed as a parameter. For all objects, the field names of an index form can be viewed in enaio® editor.

If no editor has been inserted, the name of the currently logged in user will be identified and written to the 'Editor' field (line 2).

Values are modified in the handoff file (see 'Handoff Files'). Modifications to the handoff file will only be taken into account before the action is forwarded if the return value is set to '1' (line 5, for return values see 'Client-Side Events').

Save the script by clicking  **Save**.

The next time the index form is opened, the VB script will already insert the name of the editor in case the field is still empty when saving.

If you use scripts to refer to dialog elements containing special characters, errors may occur. In this case, use internal names for referring to dialog elements.

Client-Side Events

enaio® client and enaio® server run VB scripts once the assigned event takes place. Select an event and create the VB script code. Events may need a return value which will be written into the handoff file.

Client events are divided into the following groups:

§ Application

Events related to enaio® client.

§ Data sheet

Events related to the data sheets of DMS objects which are newly created or of which the indexing is modified.

§ Hit list

Events related to hit lists which result from searches or content lists of folders and registers.

§ Query

Events related to search forms.

§ Cabinet

Event for drag & drop for moving documents into a folder

The following client-side events are available:

Event	Reference	Time of the execution	Description of the return value	Event code
AfterLogin	Application	after the user has logged on	no return value	8

Event	Reference	Time of the execution	Description of the return value	Event code
BeforeLogout	Application	before the user has logged out	1 = logout will be performed 0 = logout will not be performed	9
OnStartApp	Application	after enaio® client has been started, instantly before 'AfterLogin'	no return value	14
OnCloseApp	Application	after check in of all documents, before exiting enaio® client	1 = exit will be performed 0 = exit will not be performed	15
BeforeLink	Application	after having created a link or relation in enaio® client, before data will be passed to the server	0 = link or relation will be created 1 = relation not created, dialog will not close 2 = relation not created, dialog will close For links, every non-zero value will cancel the process.	21
AfterLink	Application	after the server has created the link or the relation	no return value	22
BeforeDeleteLink	Application	after having deleted a link or relation in enaio® client, before data will be passed to the server	0 = link or relation will be deleted -2 = process will be canceled Other values will cancel the process of deletion; it will be continued with the object selected next.	23
AfterDeleteLink	Application	after the server has deleted the link or the relation	no return value	24
StartAction	Application	Call with server notification	no return value	31
OnClickItem	Data sheet	after a button was clicked	1 = data will be imported from the handoff file to the data sheet 0 = data of the data sheet will not be modified	13

Event	Reference	Time of the execution	Description of the return value	Event code
OnShow	Data sheet	before opening the data sheet The handoff file contains the following entries: 'Action=NEW' if a data sheet is opened for new creation, 'Action=UPDATE' if a data sheet is opened for editing, 'Action=READONLY' if a data sheet is opened in read-only mode, 'Action=REQUEST' if a search form is opened.	1 = data will be imported from the handoff file to the data sheet 0 = data of the data sheet will not be modified -1 = data sheet will not be opened	1
BeforeValidate	Data sheet	after having clicked 'Save' but before enaio® client has performed the validity check and before saving The handoff file contains the following entries: 'Action=NEW' if an object is created or 'Action=UPDATE' if data are edited.	1 = data will be imported from the handoff file to the data sheet Afterwards, enaio® client performs the validity check and saves the data. 0 = the data sheet will be saved but the data of the handoff file will not be imported to the data sheet -1 = creation or modification of data will be canceled -2 = data will be imported from the handoff file to the data sheet, the data sheet will remain open, data will not be saved	2

Event	Reference	Time of the execution	Description of the return value	Event code
AfterValidate	Data sheet	after the validity check in enaio® client and before saving The handoff file contains the following entries: 'Action=NEW' if an object is created or 'Action=UPDATE' if data are edited.	1 = data will be imported from the handoff file to the data sheet 0 = the data sheet will be saved but the data of the handoff file will not be imported to the data sheet -1 = creation or modification of data will be canceled -2 = data will be imported from the handoff file to the data sheet, the data sheet will remain open, data will not be saved	36
AfterSave	Data sheet	after the validity check in enaio® client and after saving The handoff file contains the following entries: 'Action=NEW' if an object is created or 'Action=UPDATE' if data are edited.	no return value	3
BeforeCancel	Data sheet	after the 'Cancel' button was pressed	0 = the data sheet will not be closed. If other return values are used, the data sheet will be closed.	30
OnEnterPage	Data sheet	when switching the page of the 'Pagecontrol' dialog element	1 = data of fields on the page control, which are part of the handoff file, will be returned to the search form 0 = data of the handoff file will be ignored	25
OnLeavePage	Data sheet	when leaving a page of the 'Pagecontrol' dialog element	no return value	37
OnFocusGained	Data sheet	When the focus is on a text box. The event can be assigned to every text box of a data sheet.	no return value	32

Event	Reference	Time of the execution	Description of the return value	Event code
OnCellFocusGained	Data sheet	When the focus is on a table cell.	no return value	39
OnValueChanged	Data sheet	When the input in a text box was completed	0 = no change -1 = back to the text box, the entry is not changed 1 = data will be imported from the handoff file to the text box	33
OnCellValueChanged	Data sheet	When the input in a table cell was completed	0 = no change -1 = back to the text box, the entry is not changed 1 = data will be imported from the handoff file to the text box	40
BeforeAddRow	Data sheet	Before a new row is added in a table.	0 = no change -1 = row is not added	34
BeforeDeleteRow	Data sheet	Before a row is deleted in a table.	0 = no change -1 = row is not deleted	35
BeforeStartQuery	Query	after the user pressed 'Start query' on the search form, before the search is actually executed	1 = data will be imported from the handoff file to the search form enaio® client will afterwards perform the search. 0 = the search will be performed without further modification -1 = the search will be canceled	4
AfterFinishQuery	Hit list	after the search	no return value	5
BeforeDelete	Hit list	before an object is deleted	1 = deletion will be executed 0 = deletion will not be executed	11

Event	Reference	Time of the execution	Description of the return value	Event code
BeforeUndoCheckOut	Hit list	before the user action 'Undo checkout' is applied to a document in enaio® client The 'NumberOfSelectedDocuments' constant can be used to query in the script for how many documents the checkout has been undone.	-2 = the 'Undo checkout' action will not be available for this document -1 = the 'Undo checkout' action will be canceled for all selected documents 0 = the 'Undo checkout' user action will be performed 1 = the 'Undo checkout' action will be available, no user confirmation required	27
AfterDelete	Hit list	after an object was deleted	no return value	12
BeforeOpen	Hit list	before a document is opened	0 = do not open 1 = open 2 = open read-only -4 = only workflow event: opened object is shown in the document viewer	16
OnMove	Hit list	when a document or register is moved within a cabinet	0, 1 = the document or register will be moved -1 = moving is not carried out	17
OnMoveExtern	Hit list	if a document is moved to a different cabinet	0, 1 = the document or register will be moved -1 = moving is not carried out	42
OnAddLocation	Hit list	if a document or register receives another location via drag & drop	0, 1 = assign location -1 = do not assign location	18
OnCreateCopy	Hit list	when a document or register is copied	0, 1 = the document or register will be copied -1 = the document or register will not be copied	43

Event	Reference	Time of the execution	Description of the return value	Event code
BeforeSaveDocument	Hit list	before a document is checked in	0 = the document will be checked in -2 = the process will be canceled, the document will not be checked in With other return values, the document will not be checked in but it will be continued with the next document.	19
AfterSaveDocument	Hit list	after a document was checked in	no return value	20
BeforeRestore	Hit list	before an object is restored from the trash can	0 = the object is not restored If other return values are used, the object is restored.	28
AfterRestore	Hit list	after an object is restored from the trash can	no return value	29
FileDrop	Cabinet	after storing files at a location	-1 = cancellation 1 = client only updates the hit list 0 = client takes over the files	200

The 'OnContextChange' event was used for the contentviewer, a component that was replaced with the DocumentViewer in version 7.00. Therefore, this event is no longer required and deactivated in version 7.00 or higher. You can activate this event by entering a value into the `as.cfg` configuration file (\etc directory of the data directory):

```
[SYSTEM]
ENABLE_CONTEXTCHANGE_EVENT=1
```

Events for Changes in Batch Mode

The following data sheet events are integrated for changes in batch mode :

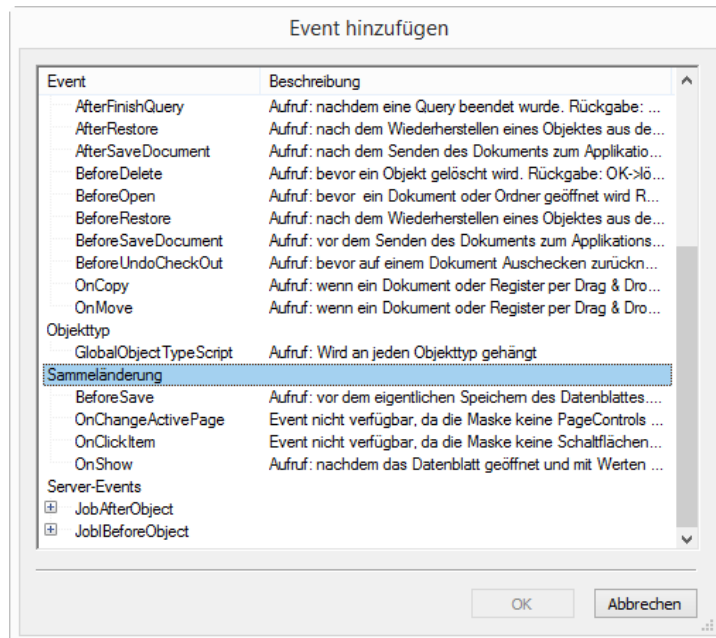
Event	Event code
BeforeValidate	100
AfterValidate	104
OnShow	101
OnClickItem	102
OnChangeActivePage	103

Handoff files contain an extra section with the following structure:

```
[BATCHUPDATE]
Count=<number of objects>
ID0=<object ID>
IDn= ...
```

Only these events can be deployed for changes in batch mode.

Events for changes in batch mode are added in the workspace using the context menu of an object type.



In the 'Object search' area, events for changes in batch mode are flagged with a lightning symbol.

The function corresponds to those of the events for an object.

Event 'StartAction'

The 'StartAction' event is invoked using the method 'krn.SendMessageToClients'. The method can be executed using the enaio® enterprise-manager or from scripts and applications. You can find information about 'krn.SendMessageToClients' in the 'enaio® Server API' documentation.

The events are invoked using the 'StartEvent' value of the 'Message' parameter. The value of the 'Text' parameter is transferred as an event file and can be evaluated by the script.

Handoff Files

enaio® client creates handoff files for events from which data can be read out and edited and into which return values can be written.

Handoff files receive the file extension *.evt and will be written to the client-side, user-specific temporary directory \temp\OSTEMP\.

The 'MsgBox filename' code in a script allows outputting the full path and the file name of the handoff file in a dialog.

You receive read and write access to the handoff files directly via the ActiveX control `oxactive.dll` (see 'The ActiveX Control OXACTIVE'). Once enaio® client has read out the required data, the handoff files will be deleted automatically.

Events may require return values to be written to the handoff files, so that enaio® client can perform the desired action. To do so, insert the following code into the handoff file:

```
ResultCode=1  
WriteToFile()
```

The following events will not create any handoff file:

- § AfterLogin
- § OnStartApp
- § AfterFinishQuery

The following events will only create an empty handoff file into which the return value needed by OS|CLIENT to continue given actions can be entered:

- § BeforeLogout
- § OnCloseApp

The following events will create handoff files with data which can be read out and further edited and into which return values can be entered:

- § OnClickItem
- § OnShow
- § BeforeValidate
- § AfterValidate
- § AfterSave
- § BeforeStartQuery
- § BeforeOpen
- § BeforeDelete
- § AfterDelete
- § OnMove
- § OnMoveExtern
- § OnAddLocation
- § OnCreateCopy
- § BeforeLink
- § BeforeDeleteLink
- § BeforeSaveDocument
- § BeforeUndoCheckOut

- § BeforeCancel
- § BeforeRestore
- § OnChangeActivePage
- § OnValueChanged
- § OnCellValueChanged
- § BeforeAddRow
- § BeforeDeleteRow

The following events will create handoff files with data which can be read out and further edited but into which a return value cannot be entered:

- § AfterLink
- § AfterDeleteLink
- § AfterSaveDocument
- § AfterRestore
- § OnLeavePage
- § OnFocusGained
- § OnCellFocusGained

If you use scripts to refer to dialog elements containing special characters, errors may occur. In this case, use internal names for referring to dialog elements.

Structure of Handoff Files

Since handoff files are text files, they can be opened with any text editor, such as Notepad.

They contain information on tabs of index forms including their data fields and contents.

Handoff files are divided into sections which are identified by the page information indicated in square brackets. The sections represent the tabs on an index form.

Commencing with zero, the pages are numbered sequentially: [PAGE00], [PAGE01], [PAGE02] etc.

Handoff files additionally include the [GLOBALS] section to which global information across all sections is saved.

If a page control was added to an index form, the pages of which will contain individual sub-sections by appending an ID to the page information:

```
[PAGE00#BD5D78C3F05A45538A226667DC644C01]
[PAGECTRL#77EEBAE5F6834B20BE5A18F939191C8A]
#NAME#=PageCtrl1
#OSINTERNNAME#=PageCtrl1
#PAGENAME1#=Run time
#PAGEOSINTERNNAME1#=Run time
#PAGEGUID1#=71134100B5BA43DD94F81CB5A100BDD6
#PAGENAME2#=Reminder
#PAGEOSINTERNNAME2#=Reminder
#PAGEGUID2#=2C8082795C704ED0A4E79D78410E6E71
...
```

The single pages of the page control can additionally be identified with the keys #PAGENAME#, #PAGEOSINTERNNAME# and #PAGEGUID#.

A key and a corresponding value are listed in each line, separated by the equals sign:

```
...
#OSMAIN#=4
#OSMODIFIED#=0
#OSFIELDMODE#=1
FULLTEXT=
VTREQUESTTYPE=0
#OSACT#=1
...
```

Type and number of keys in the handoff file depend on the calling context for both the event as well as the structure of the index form.

Keys may consist of multiple parts which are separated by separators. As a separator, either the character '\021' or ASCII code '17' must be used.

Keys of the PAGE Sections

Key	Description
#OSACT#	'1' for active tabs on the index form, otherwise '0'
#OSEXP#	If a search was started in expert mode, the key with the value '1' will be written to the handoff file.
#OSFIELDMODE#	GUID (=1) or position (=0) of the field will be expected and interpreted
#OSIDENT#	index number of the object
#OSMAIN#	main document type: '1': X document (grayscale image) '2': D document (black-and-white image) '3': P document (color image) '4': W-Document (Windows document) '5': M document (video document) '6': Q document (e-mails) '7': XML Document '0': Folder '99': Tabs
#OSMODIFIED#	time stamp denoting the last modification of an object (index form or document)
#OSNAME#	name of the tab on the index form
#OSPOS000#	sequentially numbered fields and their contents on the index form
#OSTYPE#	type ID of the object
#OSSYSTEMID#	The ID of the system that manages the object. enaio® servers always have the OSSYSTEMID '0'.
#OSFOREIGNID#	ID of an object in a third-party system, provided that the object is also managed in the third-party system.
FELD0	sequentially numbered information on the fields of the index form, separated by separators
FIELDEXT0	extension to the field 0, also numbered sequentially
FILECOUNT	number of files which are assigned to the object

Keys of the GLOBAL Section

Key	Description
Action	Available values are: UPDATE – if you have opened a data sheet for editing or modified data REQUEST – if you have opened a search form NEW – if you have opened a data sheet/object to be created

Key	Description
	READ-ONLY – if you have opened a data sheet read-only BEFORELINK – when the event of the same name is accessed
EventCode	numeric ID of the triggering event, for example '1' for the 'OnShow' event. For the numeric codes, see the tabular overview under 'Introduction'
Handle	ID of the sizing handle of the index form
OrdIdent	Internal ID of the folder
OrdType	type of the folder
RegIdent	internal ID of the register
RegType	type of the register
EXTERNDROPFILE	path to a file which is imported by dragging and dropping

Client-Side Handoff Files

The following example handoff files respectively refer to a document of the 'Log' type in the 'General' register in the 'Workflow log' folder. Folder, register and document data which will be found in the corresponding handoff files refer to these DMS objects and depend on the given event.

'Workflow log' folder form

'General' register form

'Log' document type form

OnClickItem

The event is triggered once any button of a DMS object's index form is clicked; in this example: the 'Event' button of a document of the 'Log' type.

Having triggered the `OnClickItem` event, enaio® client will create a handoff file with data on the folder, the register, the document, and on the basic parameters.

The file contains the following sections:

§ [PAGE00]

This section includes folder data, the indexing of the folder and the field definition of the folder type.

§ [PAGE01]

This section includes register data, the indexing of the register and the field definition of the register type.

§ [PAGE02]

This section includes document data, the indexing of the document and the field definition of the document type.

§ [PAGE03]

This section includes data of the basic parameters of the DMS object which the event is related.

§ [GLOBALS]

This section includes general information.

The numbering of the page sections corresponds to the order according to which the data sheets are shown from left to right in enaio® client.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=4
#OSIDENT#=882
#OSMAIN#=0
#OSMODIFIED#=1283779716
#OSPOS000#=2010
#OSPOS001#=Jobs
FILECOUNT=0
FELD0=#OSPOS000#;Year;zahl1;9;4;0;0
FIELDEXT0=#OSPOS000#08B022760BAE042CAB627F8A2945CCYearzahl1WF_Year94
00
FELD1=#OSPOS001#;Workflow family;feld1;X;50;0;0
FIELDEXT1=#OSPOS001#D70773C5AFDB4894A739C2119093DAA0Workflow
familyfeld1WF_FamilyX5000
#OSNAME#=Workflow log
[PAGE01]
#OSTYPE#=6488071
#OSIDENT#=883
#OSMAIN#=99
#OSMODIFIED#=1283779767
#OSPOS000#=April
FILECOUNT=0
FELD0=#OSPOS000#;Month;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#D622CD2C43D3490E87F36AF9ADF660D5Monthfeld1WF_Mon
thX5000
#OSNAME#=General
[PAGE02]
#OSTYPE#=131115
#OSIDENT#=884
#OSMAIN#=2
#OSMODIFIED#=1326111543
#OSFIELDMODE#=1
#OSGUID#49B4CEA9BADF49B1A3AEAC193ABC6093=Job incoming
#OSPOS000#=Job incoming
#OSGUID#DBFOC6B60C5F44A3AB37F92C8BC538A4=new
#OSPOS001#=new
#OSGUID#C5053863FA2D46B4B19C13BCAOC6D30C=324543
#OSPOS002#=324543
#OSACT#=1
FILECOUNT=0
FELD0=#OSPOS000#;Workflow name;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#049B4CEA9BADF49B1A3AEAC193ABC6Workflow
namefeld1WF_ProcessNameX5000
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000;0;0
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538ACommentfeld2WF_Co
mmentX100000
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000;0;0
FIELDEXT2=#OSPOS002#OC5053863FA2D46B4B19C13BCAOC6D3WorkflowIDfeld4WF
_ProcessIdX100000
FELD3=#OSPOS003#;Event;;K;0;0;0
FIELDEXT3=#OSPOS003#11DF4A1E9AF84A8682340807127406180EventK000
#OSNAME#=Log
#OSFOREIGNID#=0
[PAGE03]
#OSTYPE#=6553600
#OSIDENT#=884
#OSMAIN#=100
#OSMODIFIED#=0
#OSPOS000#=THOMAS
#OSPOS001#=1280227291
#OSPOS003#=ADMINISTRATOR
#OSPOS004#=1326111543
#OSPOS007#=A402C5EDF25744DBA04A622986E15042
#OSPOS006#=72
FILECOUNT=0
```

```
[GLOBALS]
EventCode=30
Action=UPDATE
Handle=67004
OrdIdent=882
OrdType=4
RegIdent=883
RegType=6488071
TargetMainType=-1
```

If, for example, the `OnClickItem` event is executed from within a search form on which the search terms 'Incoming jobs' and '324543' have been entered into the fields 'Workflow name' and 'WorkflowID', respectively, the 'Action=REQUEST' entry will be found in the [GLOBALS] section instead of the 'Action=UPDATE' entry:

```
[PAGE00]
#OSTYPE#=131115
#OSIDENT#=0
#OSMAIN#=2
#OSMODIFIED#=0
#OSFIELDMODE#=1
#OSGUID#49B4CEA9BADF49B1A3AEAC193ABC6093=Job incoming
#OSPOS000#=Job incoming
#OSGUID#DBFOC6B60C5F44A3AB37F92C8BC538A4=new
#OSPOS002#=new
#OSACT#=1
FILECOUNT=0
FELD0=#OSPOS000#;Workflow name;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#049B4CEA9BADF49B1A3AEAC193ABC6Workflow
namefeld1WF_ProcessNameX5000
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000;0;0
FIELDEXT1=#OSPOS001#DDBFOC6B60C5F44A3AB37F92C8BC538Commentfeld2WF_Co
mmentX100000
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000;0;0
FIELDEXT2=#OSPOS002#DC5053863FA2D46B4B19C13BCAOC6D3WorkflowIDfeld4WF
_ProcessIdX100000
FELD3=#OSPOS003#;Event;;K;0;0;0
FIELDEXT3=#OSPOS003#D11DF4A1E9AF84A868234080712740618DEventK000
#OSNAME#=Log
[GLOBALS]
EventCode=0
Action=REQUEST
Handle=3408762
OrdIdent=-1
OrdType=-1
RegIdent=-1
RegType=-1
```

enaio® client requires a return value for the `OnClickItem` event:

- § 'resultcode=1' will pass data from the handoff file to the index form,
- § 'resultcode=0' will not modify the data of the index form.

To close the DMS form of a DMS object by script call, use the `closedatamask` method in `OnClickItem`. The following return values are available:

- § '-1' Discard changes and close DMS form
- § '0' (Default) Do not close DMS form
- § '1' Save changes and close DMS form

The return value must be written explicitly into the handoff file.

```
WriteProfString "GLOBALS", "closedatamask", "<Return value>", OSFILE  
asfile.WriteToFile()
```

The 'Enabled=true' property can be used to activate a write-protected button on a data form with an event script or upon an `OnShow` event, to open associated documents, for example.

An event button that can be activated must exist in the event script and the data form must have no `PageControls`.

Example:

```
if AsFile.EventAction = "READONLY" then  
    MsgBox "try to activate Schaltflaeche"  
    ActivePage.ASFields.Item("&Schaltfläche").Enabled=true  
end If  
ResultCode=1  
WriteToFile()
```

From the event script, instead of the current object, an object can be shown via the object ID in the content/file preview or dashlet:

```
asfile.ContextObjIdent = ID
```

OnShow

This event can be triggered either by opening a data sheet or a search form or when creating a new DMS object.

enaio® client will create a handoff file, for example, with data on the folder [PAGE00], the register [PAGE01], the document [PAGE02], and on the basic parameters [PAGE03].

Example of the handoff file:


```
[PAGE00]
#OSTYPE#=4
#OSIDENT#=882
#OSMAIN#=0
#OSMODIFIED#=1283779716
#OSPOS000#=2010
#OSPOS001#=AdHoc V 1.02
FILECOUNT=0
FELD0=#OSPOS000#;Year;zahl1;9;4;0;0
FIELDEXT0=#OSPOS000#18A1529714954419924950FFFD930FDEYearzahl1WF_Year
9400
FELD1=#OSPOS001#;Workflow family;feld1;X;50;0;0
FIELDEXT1=#OSPOS001#8B022760BAE042CAB627F8A2945CC044Workflow
familyfeld1WF_FamilyX5000
#OSNAME#=Workflow log
[PAGE01]
#OSTYPE#=6488071
#OSIDENT#=883
#OSMAIN#=99
#OSMODIFIED#=1283779767
#OSPOS000#=April
FILECOUNT=0
FELD0=#OSPOS000#;Month;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#D622CD2C43D3490E87F36AF9ADF660D5Monthfeld1WF_Mon
thX5000
#OSNAME#=General
[PAGE02]
#OSTYPE#=131115
#OSIDENT#=884
#OSMAIN#=2
#OSMODIFIED#=1326113515
#OSFIELDMODE#=1
#OSGUID#49B4CEA9BADF49B1A3AEAC193ABC6093=Job incoming
#OSPOS000#=Job incoming
#OSGUID#DBFOC6B60C5F44A3AB37F92C8BC538A4=new
#OSPOS001#=new
#OSGUID#C5053863FA2D46B4B19C13BCAOC6D30C=324543
#OSPOS002#=324543
#OSACT#=1
FILECOUNT=0
FELD0=#OSPOS000#;Workflow name;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Workflow
namefeld1WF_ProcessNameX5000
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000;0;0
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Commentfeld2WF_C
ommentX100000
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000;0;0
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CWorkflowIDfeld4W
F_ProcessIdX100000
FELD3=#OSPOS003#;Event;;K;0;0;0
FIELDEXT3=#OSPOS003#11DF4A1E9AF84A868234080712740618EventK000
#OSNAME#=Log
#OSFOREIGNID#=0
[PAGE03]
#OSTYPE#=6553600
#OSIDENT#=884
#OSMAIN#=100
#OSMODIFIED#=0
FILECOUNT=0
[GLOBALS]
EventCode=1
Action=UPDATE
Handle=1442014
OrdIdent=882
OrdType=4
```

```
RegIdent=883
RegType=6488071
CANRESETFIELDS=1
CANINSERTFIELDS=1
```

The `OnShow` event is also triggered by creating a new DMS object. The [GLOBALS] section will then include the 'Action=NEW' entry. When creating the document by dragging and dropping, the 'EXTERNDROPPFILE' entry will indicate the path to the file. If the DMS object's search form is opened, the 'Action=REQUEST' entry will be added. When opening the data sheet in read-only mode, the 'Action=READONLY' entry will be added. Scripts which do not take account of several action modes will lead to errors.

If an object is created by copying, the handoff file contains an additional entry with the ID of the source object. The entry has the following structure: 'CopyFrom=ID'

If an object from a register is opened, the handoff file contains the location information. The entry has the following structure:

```
[GLOBALS]
EventLocationFolderIdent=81
EventLocationFolderType=0
EventLocationRegisterIdent=293
EventLocationRegisterType=6488065
```

If an object from a first level folder is opened, the location information contained in the handoff file has the following structure:

```
[GLOBALS]
EventLocationFolderIdent=81
EventLocationFolderType=0
EventLocationRegisterIdent=4294967295
EventLocationRegisterType=4294967295
```

If an object from a location which is not a folder is opened, the handoff file has the following structure:

```
[GLOBALS]
EventLocationFolderIdent=4294967295
EventLocationFolderType=4294967295
EventLocationRegisterIdent=4294967295
EventLocationRegisterType=4294967295
```

4294967295 means 'not specified'.

enaio® client requires a return value for the `OnShow` event:

'resultcode=1' will pass data from the handoff file to the index form,

'resultcode=0' will not modify the data,

'resultcode=-1' will cancel the opening process of the index form.

The `OnShow` event, for example, allows editing of the 'Dialog element visible' property:

```
Activepage.AsFields.Item("Comment").Visible = false
ResultCode=1
WriteToFile()
```

In the same way, the 'Enabled=False' property activates the write-protection of certain fields.

Write-protected buttons on a data form or for an `OnShow` event can also be activated with the property to open associated documents, for example.

An event button that can be activated must exist in the event script and the data form must have no `PageControls`.

Example:

```
if AsFile.EventAction = "READONLY" then
    MsgBox "try to activate Schaltflaeche"
    ActivePage.AsFields.Item("&Schaltfläche").Enabled=true
end If
ResultCode=1
WriteToFile()
```

If a data sheet was opened by use of the 'Quickfinder' AddOn, the handoff file will contain the following data in the [GLOBALS] section:

QUICKFINDER=1	Indicates that the data sheet has been opened by use of the 'Quickfinder' AddOn.
QUICKPARENTOBJECT=Object type ID	Specifies the ID of the document type through which the 'Quickfinder' AddOn has been executed.

The items 'CANRESETFIELDS=1' and 'CANINSERTFIELDS=1' in [GLOBALS] section relate to the context menu functions 'Reset' and 'Paste' on a form. 'Reset' clears all fields, 'Paste' inserts copied entries of another object.

For these functions not to be available, set their value to '0'. This will disable the options.

From the event script, instead of the current object, an object can be shown via the object ID in the content/file preview or dashlet:

```
asfile.ContextObjIdent = ID
```

BeforeValidate

The event is triggered when the indexing is changed by clicking the 'Save' button on the index form of a DMS object; in the example: a document of the 'Log' type. When a newly created DMS object is saved, the event will also be triggered and, instead of the 'Action=UPDATE' entry, the 'Action=NEW' entry will be added to the [GLOBALS] section.

enaio® client will create a handoff file with data on the folder [PAGE00], the register [PAGE01], the document [PAGE02], and on the basic parameters [PAGE03].

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=4
#OSIDENT#=882
#OSMAIN#=0
#OSMODIFIED#=1283779767
#OSPOS000#=2010
#OSPOS001#=Jobs
FILECOUNT=0
FELD0=#OSPOS000#;Year;zahl1;9;4;0;0
FIELDEXT0=#OSPOS000#18A1529714954419924950FFFD930FDEYearzahl1WF_Year
9400
FELD1=#OSPOS001#;Workflow family;feld1;X;50;0;0
FIELDEXT1=#OSPOS001#8B022760BAE042CAB627F8A2945CC044Workflow
familyfeld1WF_FamilyX5000
#OSNAME#=Workflow log
[PAGE01]
#OSTYPE#=6488071
#OSIDENT#=883
#OSMAIN#=99
#OSMODIFIED#=1283779767
#OSPOS000#=April
FILECOUNT=0
FELD0=#OSPOS000#;Month;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#D622CD2C43D3490E87F36AF9ADF660D5Monthfeld1WF_Mon
thX5000
#OSNAME#=General
[PAGE02]
#OSTYPE#=131115
#OSIDENT#=884
#OSMAIN#=2
#OSMODIFIED#=1326113973
#OSFIELDMODE#=1
#OSGUID#49B4CEA9BADF49B1A3AEAC193ABC6093=Job incoming
#OSPOS000#=Job incoming
#OSGUID#DBFOC6B60C5F44A3AB37F92C8BC538A4=new and edited
#OSPOS001#=new and edited
#OSGUID#C5053863FA2D46B4B19C13BCAOC6D30C=324543
#OSPOS002#=324543
#OSACT#=1
FILECOUNT=1
FELD0=#OSPOS000#;Workflow name;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Workflow
namefeld1WF_ProcessNameX5000
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000;0;0
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Commentfeld2WF_C
ommentX100000
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000;0;0
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CWorkflowIDfeld4W
F_ProcessIdX100000
FELD3=#OSPOS003#;Event;;K;0;0;0
FIELDEXT3=#OSPOS003#11DF4A1E9AF84A868234080712740618EventK000
#OSNAME#=Log
#OSFOREIGNID#=0
[PAGE03]
#OSTYPE#=6553600
#OSIDENT#=884
#OSMAIN#=100
#OSMODIFIED#=0
#OSPOS000#=THOMAS
#OSPOS001#=1280227291
#OSPOS003#=ADMINISTRATOR
#OSPOS004#=1326113973
#OSPOS007#=A402C5EDF25744DBA04A622986E15042
#OSPOS006#=72
FILECOUNT=0
```

```
[GLOBALS]
EventCode=2
Action=UPDATE
Handle=590516
OrdIdent=882
OrdType=4
RegIdent=883
RegType=6488071
TargetMainType=-1
```

If an object is created by copying, the handoff file contains an additional entry with the ID of the source object. The entry has the following structure: 'CopyFrom=ID'

enaio® client requires a return value for the event:

'resultcode=1' will save data from the handoff file in the index form.

'resultcode=0' Data will not be changed.

'resultcode=-1' will not apply any data and will keep the index form open.

'resultcode=-2' will pass data from the handoff file to the index form, the index form with the data changes will remain open in enaio® client.

enaio® client checks the data and, with 'resultcode=1', will leave the index form open and show respective notices to the user if any data does not fulfill the requirements.

The catalog check can be switched off. To do so, write the `CHECKCATALOGVALUES=0` entry into the [GLOBALS] section:

```
oxhelp.writeprofstring "GLOBALS", "CHECKCATALOGVALUES", 0, osfile
Resultcode = 1
WriteToFile()
```

AfterValidate

The event matches BeforeValidate but is executed after the validity check of enaio® client.

enaio® client requires a return value for the event:

'resultcode=1' will save data from the handoff file in the index form without further validation by enaio® client.

'resultcode=0' Data will not be changed.

AfterSave

The event will be executed after the data of a data sheet is saved. enaio® client will not read the data from the handoff file and no return value is expected.

The handoff file includes data on the folder [PAGE00], the document [PAGE01], and on the basic parameters [PAGE02]. The document is not located in any register.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=4
#OSIDENT#=882
#OSMAIN#=0
#OSMODIFIED#=1283779716
#OSPOS000#=2010
#OSPOS001#=Jobs
FILECOUNT=0
FELD0=#OSPOS000#;Year;zahl1;9;4;0;0
FIELDEXT0=#OSPOS000#18A1529714954419924950FFFD930FDEYearzahl1WF_Year
9400
FELD1=#OSPOS001#;Workflow family;feld1;X;50;0;0
FIELDEXT1=#OSPOS001#8B022760BAE042CAB627F8A2945CC044Workflow
familyfeld1WF_FamilyX5000
#OSNAME#=Workflow log
[PAGE01]
#OSTYPE#=6488071
#OSIDENT#=883
#OSMAIN#=99
#OSMODIFIED#=1283779767
#OSPOS000#=April
FILECOUNT=0
FELD0=#OSPOS000#;Month;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#D622CD2C43D3490E87F36AF9ADF660D5Monthfeld1WF_Mon
thX5000
#OSNAME#=General
[PAGE02]
#OSTYPE#=131115
#OSIDENT#=884
#OSMAIN#=2
#OSMODIFIED#=1326114350
#OSFIELDMODE#=1
#OSGUID#49B4CEA9BADF49B1A3AEAC193ABC6093=Job incoming
#OSPOS000#=Job incoming
#OSGUID#DBFOC6B60C5F44A3AB37F92C8BC538A4=new and edited
#OSPOS001#=new and edited
#OSGUID#C5053863FA2D46B4B19C13BCAOC6D30C=324543
#OSPOS002#=324543
#OSACT#=1
FILECOUNT=1
FELD0=#OSPOS000#;Workflow name;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Workflow
namefeld1WF_ProcessNameX5000
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000;0;0
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Commentfeld2WF_C
ommentX100000
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000;0;0
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CWorkflowIDfeld4W
F_ProcessIdX100000
FELD3=#OSPOS003#;Event;;K;0;0;0
FIELDEXT3=#OSPOS003#11DF4A1E9AF84A868234080712740618EventK000
#OSNAME#=Log
#OSFOREIGNID#=0
[PAGE03]
#OSTYPE#=6553600
#OSIDENT#=884
#OSMAIN#=100
#OSMODIFIED#=0
#OSPOS000#=THOMAS
#OSPOS001#=1280227291
#OSPOS003#=ADMINISTRATOR
#OSPOS004#=1326114350
#OSPOS007#=A402C5EDF25744DBA04A622986E15042
#OSPOS006#=72
FILECOUNT=0
```

```
[GLOBALS]
EventCode=2
Action=UPDATE
Handle=458922
OrdIdent=882
OrdType=4
RegIdent=883
RegType=6488071
TargetMainType=-1
```

BeforeStartQuery

The event will be executed once the 'Start query' function is started from within a search form. The handoff file includes data on the search forms. This data can be changed by using the VB script.

enaio® client will pass the data from the handoff file to the search forms and start the search provided that the 'resultcode=1' entry was written into the handoff file.

'resultcode=0' will start the search without further modification.

'resultcode=-1' will cancel the search.

The handoff file includes data on the search forms and entered search terms. In the example, the search terms '324543' and '2010' have been entered into the fields 'WorkflowID' on the search form of the document and 'Year' on the search form for folders, respectively.

The numbering of the page sections of combined queries corresponds to the order according to which the data sheets are shown from left to right in enaio® client. The event is assigned to a DMS object and will only be run if the data sheet of the DMS object is active when starting the combined query.

Example of the handoff file:


```
[ PAGE00 ]
#OSTYPE#=131115
#OSIDENT#=0
#OSMAIN#=2
#OSMODIFIED#=0
#OSFIELDMODE#=1
#OSGUID#C5053863FA2D46B4B19C13BCAOC6D30C=324543
#OSPOS002#=324543
#OSACT#=1
FILECOUNT=0
FELD0=#OSPOS000#;Workflow name;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Workflow
namefeld1WF_ProcessNameX5000
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000;0;0
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Commentfeld2WF_C
ommentX100000
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000;0;0
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CWorkflowIDfeld4W
F_ProcessIdX100000
FELD3=#OSPOS003#;Event;;K;0;0;0
FIELDEXT3=#OSPOS003#11DF4A1E9AF84A868234080712740618EventK000
#OSNAME#=Log
[ PAGE01 ]
#OSTYPE#=6488071
#OSIDENT#=884
#OSMAIN#=99
#OSMODIFIED#=1283779716
#OSFIELDMODE#=1
FILECOUNT=0
FELD0=#OSPOS000#;Month;feld1;X;50;0;0
FIELDEXT0=#OSPOS000#D622CD2C43D3490E87F36AF9ADF660D5Monthfeld1WF_Mon
thX5000
#OSNAME#=General
[ PAGE02 ]
#OSTYPE#=131115
#OSIDENT#=884
#OSMAIN#=2
#OSMODIFIED#=1326114350
#OSFIELDMODE#=1
#OSGUID#DDF9D11554884EA6BA217F99189CDF01=2010
#OSPOS000#=2010
FILECOUNT=0
FELD0=#OSPOS000#;Year;zahl1;9;4;0;0
FIELDEXT0=#OSPOS000#DDF9D11554884EA6BA217F99189CDF01Jahrzahl1WF_Year
9400
FELD1=#OSPOS001#;Workflow family;feld1;X;50;0;0
FIELDEXT1=#OSPOS001#D70773C5AFDB4894A739C2119093DAA0Workflow
familyfeld1WF_FamilyX5000
#OSNAME#=Workflow log
[ GLOBALS ]
EventCode=4
Action=
Handle=1905136
OrdIdent=-1
OrdType=-1
RegIdent=-1
RegType=-1
```

BeforeOpen

The event will be executed after a document was opened. The handoff file includes data on the indexing of the document but no further data, such as the basic parameters.

enaio® client will open the document provided that the 'resultcode=1' entry was written to the handoff file.

'resultcode=0' will not open the document. The document is also not opened without a return value.

'resultcode=-4' contains the object ID and the object type as the return value. If you switch between different workflow processes, hit lists, and opened locations, the document will still be displayed in the DocumentViewer.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=131115
FILECOUNT=1
#OSPOS000#=Job incoming
FELD0=#OSPOS000#;Workflow name;feld1;X;50
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Prozessnamefeld1
WF_ProcessNameX50
#OSPOS001#=new and edited
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Bemerkungfeld2WF
_CommentX1000
#OSPOS002#=324543
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CProzessIdfeld4WF
_ProcessIdX1000
#OSIDENT#=884
#OSNAME#=Log
[GLOBALS]
EventCode=16
Action=READONLY
OrdIdent=882
OrdType=4
RegIdent=883
RegType=6488071
CHECKOUT=0
```

BeforeDelete

The event will be executed after a document, folder, or register was deleted. The handoff file includes data on the indexing of the DMS object but no further data, such as the basic parameters.

enaio® client will delete the document provided that the 'resultcode=1' entry was written to the handoff file.

'resultcode=0' will not delete the document.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=131115
FILECOUNT=1
#OSPOS000#=Job incoming
FELD0=#OSPOS000#;Workflow name;feld1;X;50
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Workflow
namefeld1WF_ProcessNameX50
#OSPOS001#=copied
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Commentfeld2WF_C
ommentX1000
#OSPOS002#=324543
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CWorkflowIDfeld4W
F_ProcessIdX1000
#OSIDENT#=2189
#OSNAME#=Log
[GLOBALS]
EventCode=11
OrdIdent=0
OrdType=4
```

BeforeUndoCheckOut

The event will be executed after the checkout of one or more documents was undone. The handoff file includes data on the indexing of the document but no further data, such as the basic parameters.

'resultcode=1' will undo the checkout without user confirmation.

enaio® client will undo the checkout of the document provided that the 'resultcode=0' entry was written to the handoff file.

If more than one document is selected, 'resultcode=-1' will not undo the checkout of these documents.

'resultcode=-2' will not undo the checkout of the current document.

The 'NumberOfSelectedDocuments' constant can be used to query in the script for how many documents the checkout has been undone.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=131115
FILECOUNT=1
#OSPOS000#=Job incoming
FELD0=#OSPOS000#;Workflow name;feld1;X;50
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Workflow
namefeld1WF_ProcessNameX50
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Commentfeld2WF_C
ommentX1000
#OSPOS002#=324543
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CWorkflowIDfeld4W
F_ProcessIdX1000
#OSIDENT#=2189
#OSNAME#=Log
[GLOBALS]
EventCode=27
OrdIdent=882
OrdType=4
RegIdent=883
RegType=6488071
```

AfterDelete

The event will be executed after a document, folder, or register was deleted. The handoff file includes data on the indexing of the DMS object but no further data, such as the basic parameters.

enaio® client will create the handoff file and run the VB script but will not read the data from the handoff file.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=4
FILECOUNT=0
#OSPOS000#=2010
FELD0=#OSPOS000#;Year;zahl1;9;4
FIELDEXT0=#OSPOS000#18A1529714954419924950FFFD930FDEYearzahl1WF_Year
9400
FELD1=#OSPOS001#;Workflow family;feld1;X;50
FIELDEXT1=#OSPOS001#8B022760BAE042CAB627F8A2945CC044Workflow
familyfeld1WF_FamilyX5000
#OSIDENT#=1805
#OSNAME#=Workflow log
[GLOBALS]
EventCode=12
```

OnMove

The event will be executed when moving a DMS object within a cabinet to another location. The handoff file includes data on the indexing of the DMS object.

The [MOVEINFO] section will include information on the source and target location. A VB script may be used, for example, in order to check and change the target location.

§ SOURCEINFO

```
SOURCEINFO=FolderID,FolderType,RegisterID,RegisterType
```

§ DESTINFO

```
DESTINFO=FolderID,RegisterID
```

The object will be moved according to the data in the handoff file provided that the 'resultcode=1' entry was written to the handoff file.

'resultcode=-1' will not move the document.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=131115
FILECOUNT=1
#OSPOS000#=Job incoming
FELD0=#OSPOS000#;Workflow name;feld1;X;50
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Prozessnamefeld1
WF_ProcessNameX50
#OSPOS001#=new order January
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Bemerkungfeld2WF
_CommentX1000
#OSPOS002#=334543
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CProzessIdfeld4WF
_ProcessIdX1000
#OSIDENT#=2196
#OSNAME#=Log
[GLOBALS]
EventCode=17
OrdIdent=2190
OrdType=4
RegIdent=0
RegType=0
[MOVEINFO]
SOURCEINFO=2190,4,0,0
DESTINFO=2190,2193
DESTINFO2=2190,2193,6488071
```

OnMoveExtern

The event will be executed when moving a DMS object to another cabinet. The handoff file includes data on the indexing of the DMS object and matches the event 'OnMove'.

OnAddLocation

The event will be executed when a register or document receives another location via drag & drop.

The handoff file includes data on the indexing of the DMS object.

The [COPYINFO] section will include information on the source and target location. A VB script may be used, for example, in order to check and change the target location.

§ SOURCEINFO

```
SOURCEINFO=FolderID,FolderType,RegisterID,RegisterType
```

§ DESTINFO

```
DESTINFO=FolderID,RegisterID
```

A location is added with the data from the handoff file, provided that the 'resultcode=1' entry was written to the handoff file.

'resultcode=-1' does not add a location.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=131115
FILECOUNT=1
#OSPOS000#=order processing
FELD0=#OSPOS000#;Workflow name;feld1;X;50
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Workflow
namefeld1WF_ProcessNameX50
#OSPOS001#=order will now be processed further
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000
FIELDEXT1=#OSPOS001#DBFOC6B60C5F44A3AB37F92C8BC538A4Commentfeld2WF_C
ommentX1000
#OSPOS002#=3345439
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCAOC6D30CWorkflowIDfeld4W
F_ProcessIdX1000
#OSIDENT#=2197
#OSNAME#=Log
[GLOBALS]
EventCode=18
OrdIdent=2190
OrdType=4
RegIdent=2194
RegType=6488071
[ COPYINFO]
SOURCEINFO=2190,4,2194,6488071
DESTINFO=2190,2195
DESTINFO2=2190,2195,6488071
```

OnCreateCopy

The event will be executed when a register or document is copied. The handoff file includes data on the indexing of the DMS object and matches the event 'OnAddLocation'.

BeforeLink

The event will be executed when creating a relation or a link through the notes window or the notes area of a folder window.

The handoff file includes the data on the indexing of both DMS objects in the sections [PAGE00] and [PAGE01].

In the [GLOBALS] section the referenced objects are specified as follows:

```
[GLOBALS]
EventCode=21
Action=BEFORELINK
LINKOBJECTID1=1868
LINKOBJECTTYPE1=4
LINKOBJECTID2=1808
LINKOBJECTTYPE2=4
```

If relations were created, the relation data would be specified as well.

'resultcode=2' will not create the relation and the relation dialog will close.

For relations, 'resultcode=1' will not create the relation but the relation dialog will not be closed either.

Links and relations will be created provided that the 'resultcode=0' entry was written to the handoff file.

For links, every non-zero value will cancel the process. The link will not be created.

BeforeDeleteLink

The event will be executed when deleting a relation or a link through the notes window or the notes area of a folder window.

The handoff file corresponds to the one which is created by the 'BeforeLink' event.

Links and relations will be deleted provided that the 'resultcode=0' entry was written to the handoff file.

Other values will cancel the process and the link or relation will not be deleted.

AfterLink

The event will be executed after a link or relation was created.

The handoff file corresponds to the one which is created by the 'BeforeLink' event. enaio® client will not return any data.

AfterDeleteLink

The event will be executed after a link or relation was deleted.

The handoff file corresponds to the one which is created by the 'BeforeLink' event. enaio® client will not read any data from the handoff file.

BeforeSaveDocument

The event will be executed before a document is checked in.

Checked out documents will also be checked in when enaio® client is exited. The event will also be executed.

In the [PAGE00] section the handoff file includes the indexing data of the document. The name and path of the document files in the local cache area of the workstation are specified as well.

Example:

```
FILE0=...\LOKALE~1\TEMP\OSTEMP\00000791\CACHE\03\11\2D\00000C2D.000
```

```
FILE1=...\LOKALE~1\TEMP\OSTEMP\00000791\CACHE\03\11\2D\00000C2D.001
```

The [GLOBALS] section additionally includes the ID and the type of both the folder and the register.

The document will be checked in, provided that the 'resultcode=0' entry was written to the handoff file.

'resultcode=-2' will cancel the process.

With other return values, the current document will not be checked in but it will be continued with the next document.

AfterSaveDocument

The event will be executed after a document is checked in.

The handoff file corresponds to the one which is created by the 'BeforeSaveDocument' event. It neither includes the names nor the paths of the document files; if the file was transferred by dragging and dropping, the source path to this file is indicated as a value of the 'EXTERNDROPPFILE' entry in the handoff file.

enaio® client will not read the data from the handoff file.

BeforeCancel

The event is triggered by clicking on the **Cancel** button on a non-write-protected data sheet.

The handoff file corresponds to the one which is created by the 'OnShow' event.

Set 'resultcode' to '0' to have the data sheet kept open.

With other return values the data sheet will be closed.

BeforeRestore

The event is triggered by selecting an object from the trash can and clicking the **Restore** button.

'resultcode=0' will have the restoring process canceled.

With all other return values the object will be restored.

The handoff file includes a 'Page' section with the object's data. The 'Global' section may specify the folder and register of the former filing location through the ID.

Example of the handoff file:

```
[PAGE00]
#OSTYPE#=131115
FILECOUNT=1
#OSPOS000#=Job incoming
FELD0=#OSPOS000#;Workflow name;feld1;X;50
FIELDEXT0=#OSPOS000#49B4CEA9BADF49B1A3AEAC193ABC6093Prozessnamefeld1
WF_ProcessNameX50
#OSPOS001#=not archived
FELD1=#OSPOS001#;Bemerkung;feld2;X;1000
FIELDEXT1=#OSPOS001#DBF0C6B60C5F44A3AB37F92C8BC538A4Bemerkungfeld2WF
_CommentX1000#OSPOS002#=3827982
FELD2=#OSPOS002#;WorkflowID;feld4;X;1000
FIELDEXT2=#OSPOS002#C5053863FA2D46B4B19C13BCA0C6D30CProzessIdfeld4WF
_ProcessIdX1000
#OSIDENT#=888
#OSNAME#=Log
[GLOBALS]
EventCode=28
OrdIdent=882
OrdType=4
RegIdent=883
RegType=6488071
```

AfterRestore

The event is triggered as soon as an object is restored from the trash can.

The handoff file corresponds to the one which is created by the 'BeforeRestore' event.

Return values are ignored.

OnChangeActivePage

The event will be executed after another page of the 'Page control' dialog element was activated.

Values will not be returned. When switching to a search form, the [Globals] section will include the `Action=REQUEST` entry, whereas when switching to a data sheet, it will include the `Action=UPDATE` entry.

The following extract shows the general structure of the handoff file:

```
[ PAGE00 ]
#OSTYPE#=3
#OSIDENT#=0
#OSMAIN#=0
#OSMODIFIED#=0
#OSFIELDMODE#=1
FULLTEXT=
VTREQUESTTYPE=0
#OSACT#=1
FILECOUNT=0
FELD0=#OSPOS000#;SUPFEST;feld22;X;3;0;1
FIELDEXT0=#OSPOS000#12DD22481B2C45848E5BC0CDE959F271SUPFESTfeld22SUP
FESTX301
FELD1=#OSPOS001#;PageCtrl27;;C;0;0;0
FIELDEXT1=#OSPOS001#65DD459E62BE4DE99932ED6EA9753438PageCtrl27C000
#OSNAME#=Customer
[ PAGE00#BD5D78C3F05A45538A226667DC644C01 ]
#OSTYPE#=3
#OSFIELDMODE#=1
#OSGUID#6BE24A2D207F48FF8CD7965EB840035C=
#OSPOS000#=
#OSGUID#821A6F059F02418F875FB6F2D3496694=
#OSPOS001#=
#OSGUID#173CFBCFBF244B03B0645B005F11F5E0=
#OSPOS002#=
#OSGUID#910C438E76534801B66C12DDDDF73353=
#OSPOS003#=
#OSGUID#7C68B08185704C1DB1AD7C88B17CF529=
#OSPOS004#=
#OSGUID#58FCCDA78AA44B6093B8B2940531E835=
#OSPOS005#=
#OSGUID#8914CE6CE13942D3A6BCFFC6DF1627C2=
#OSPOS006#=
#OSGUID#BD1779FB5DD7496C9A79F91EF4A71517=
#OSPOS007#=
#OSGUID#CE5DFE7C5006443BBC2E0ACF994890AC=
#OSPOS008#=
#OSGUID#AA808EFBA7F04905A48092E30FF92403=
#OSPOS009#=
#OSGUID#EFD8FBAF228241BF983E4F1973375CE7=
#OSPOS010#=
#OSGUID#6FBFBC01EBAC406A8A048997F3C33351=
#OSPOS011#=
#OSACT#=0
#OSENABLED#=1
#FIELDPOS#=#OSPOS001#
#OSNAME#=Customer data
FELD0=#OSPOS000#;Company name;feld1;X;60;0;0
FIELDEXT0=#OSPOS000#6BE24A2D207F48FF8CD7965EB840035CFirmennamefeld1C
ompany nameX6000
FELD1=#OSPOS001#;Additional;feld2;X;60;0;0
FIELDEXT1=#OSPOS001#821A6F059F02418F875FB6F2D3496694Zusatzfeld2Addit
ionalX6000
FELD2=#OSPOS002#;Street;feld3;X;40;0;0
FIELDEXT2=#OSPOS002#173CFBCFBF244B03B0645B005F11F5E0Straßefeld3Stree
tX4000
FELD3=#OSPOS003#;Land;feld4;A;3;0;0
FIELDEXT3=#OSPOS003#910C438E76534801B66C12DDDDF73353Landfeld4LandA30
0
FELD4=#OSPOS004#;Zip code;feld5;Z;5;0;0
FIELDEXT4=#OSPOS004#7C68B08185704C1DB1AD7C88B17CF529PLZfeld5PLZZ500
FELD5=#OSPOS005#;City;feld6;X;30;0;0
FIELDEXT5=#OSPOS005#58FCCDA78AA44B6093B8B2940531E835Ortfeld6OrtX3000
FELD6=#OSPOS006#;State;feld7;X;50;0;0
```

```

FIELDEXT6=#OSPOS006#8914CE6CE13942D3A6BCFFC6DF1627C2Bundeslandfeld7B
undeslandX5000
FELD7=#OSPOS007#;Phone;feld8;X;20;0;0
FIELDEXT7=#OSPOS007#BD1779FB5DD7496C9A79F91EF4A71517Telefonfeld8Phon
eX2000
FELD8=#OSPOS008#;Fax;feld9;X;20;0;0
FIELDEXT8=#OSPOS008#CE5DFE7C5006443BBC2E0ACF994890ACFaxfeld9FaxX2000
FELD9=#OSPOS009#;E-Mail;feld10;X;80;0;0
FIELDEXT9=#OSPOS009#AA808EFBA7F04905A48092E30FF92403E-Mailfeld10E-
Mail_addressX8000
FELD10=#OSPOS010#;Internet;feld11;X;80;0;0
FIELDEXT10=#OSPOS010#EFD8FBAF228241BF983E4F1973375CE7Internetfeld11I
nternetX8000
FELD11=#OSPOS011#;Class;feld29;X;30;0;0
FIELDEXT11=#OSPOS011#6FBFBC01EBAC406A8A048997F3C33351Klassefeld29Cla
ssX3000
[PAGE00#BD85F472CDF646CC82245ECCF4D03EBE]
#OSTYPE#=3
#OSFIELDMODE#=1
#OSGUID#328DD5567B05420893A94090FAE565B0=
#OSPOS000#=
#OSGUID#53A47363F74D4BCD8C944EBD502BE725=
#OSPOS001#=
#OSGUID#EF5B3ACFA2FF418A98048CBECFB2F337=
#OSPOS002#=
#OSGUID#2ED784B358B04467A969EBBCC1995ADC=
#OSPOS003#=
#OSGUID#0003EDBB81654525A9638298CCC4F546=
#OSPOS004#=
#OSGUID#9A2D322CEA694684A0AC210B05F6B848=
#OSPOS005#=
#OSGUID#9504BB059BDB419AAF5EC49F59BF0B=
#OSPOS006#=
#OSGUID#136F0270745A4BE480EE89DB0CD4376E=
#OSPOS007#=
#OSACT#=1
#OSENABLED#=1
#FIELDPOS#=#OSPOS001#
#OSNAME#=Description
FELD0=#OSPOS000#;Industry;feld15;X;20;0;0
FIELDEXT0=#OSPOS000#328DD5567B05420893A94090FAE565B0Branchefeld15Bra
ncheX2000
FELD1=#OSPOS001#;Origin;feld16;X;20;0;0
...

```

The #OSACT#=1 entry indicates the active page.

The PAGE00, into which the page control was inserted, will be divided into the defined pages of the page control which will receive an individual ID:

```

§ [PAGE00#BD5D78C3F05A45538A226667DC644C01]
§ [PAGE00#BD85F472CDF646CC82245ECCF4D03EBE]
§ ...

```

FileDrop

The event is executed after storing files at a location via drag & drop.

'resultcode=-1' cancels the action and does not adopt any files.

'resultCode=1' will only refresh the hit list.

'resultcode=0' will store the specified data in the handoff file via enaio® client.

Example of the handoff file:

```
[GLOBALS]
EventCode=200
OrdIdent=673
OrdType=3
RegIdent=967
RegType=6488070
[FILES]
COUNT=2
FILE1=C:\Users\thomas\Documents\Auftragsbest.tif
FILE2=C:\Users\thomas\Documents\Schreiben.tif
```

Handoff Data of Tables

Data contained in tables of index forms will be written to the handoff file as follows:

```
[object75list1]
ZEILE0={1234Verkauf567842,13333,522}
ZEILEDB0='1234','Sales','5678','42,13','3','33,5','22'
ZEILE1={5678Verkauf356432,673,545,233}
ZEILEDB1='5678','Verkauf','3564','32,67','3,5','45,2','33'
REQFIELDS=feld1X50,feld2X50,feld3X5,feld4X20,feld5X10,feld6X20,feld7X20
FIELDS=feld1,feld2,feld3,feld4,feld5,feld6,feld7
[LISTCONTROL]
TABLES=object75list1
```

Table of the example file:

Line	Einteilung	Funktion	Anzahl	Listenpreis	Rabatt	Einzelpreis	Betrag
1	1234	Verkauf	5678	42.13	3	33.5	22

Table of the example file with a selection catalog:

Line	Einteilung	Funktion	Anzahl	Listenpreis	Rabatt	Einzelpreis	Betrag
1	1234	Verkauf	5678	42.13	3	33.5	22
					3		
					10		
					15		

Example of a modification (writing the values of the first line of the table):

```

a = "{1.Wert" & chr(17) & "2.Wert" & chr(17) & "3.Wert" & chr(17) _
& "4.Wert" & chr(17) & "5.Wert" & chr(17) & "6.Wert" & chr(17) _
& "7.Wert}"
oxhelp.writeprofstring "object75list1", "zeile0", a, osfile

```

These modifications will be written provided that the 'resultcode=1' entry was written to the handoff file.

The ActiveX Control OXACTIVE.DLL

The ActiveX control `oxactive.dll` offers methods and objects that allow reading and editing data of the handoff file and the configuration file.

During the installation of enaio®, the control will be installed and registered automatically.

Through the VB Script AddOn, the control can be directly used in the VB script editor. In other environments, the ActiveX control `oxactive.dll` is integrated as follows:

CoxHelp Interface in support of the VB script programming

Object creation on VBScript:

```

Dim x
Set x = CreateObject('oxactive.CoxHelp')

```

Methods

GetProfString()

```

HRESULT GetProfString(BSTR bstrSec, BSTR bstrKey, BSTR bstrDefault,
VARIANT* pvarReturn, BSTR bstrFile)

```

This function provides the functionality of the Windows API function

`GetPrivateProfileString`.

Input:	BSTR bstrSec	the section name in the handoff file
	BSTR bstrKey	the key name in this section
	BSTR bstrDefault	default return value if no value can be determined
	BSTR bstrFile	file name of the handoff file
Output:	VARIANT* pVarReturn	determines key value as string

Example:

```

oxhelp.GetProfString "PAGE00", "#OSEXP#", -1, rvalue, osfile

```

If the value cannot be determined, the value of the '#OSEXP#' key in the 'Page00' section is written to the 'rvalue' variable, otherwise '-1'.

WriteProfString()

```
HRESULT WriteProfString(BSTR bstrSection, BSTR bstrKey, BSTR
bstrValue, BSTR bstrFile)
```

This function provides the functionality of the Windows API function WritePrivateProfileString.

Input:

BSTR bstrSection	the section name in the handoff file
BSTR bstrKey	the key name in the handoff file
BSTR bstrValue	the key value
BSTR bstrFile	path and file name of the handoff file

Output: -

Before the `oxhelp.WriteProfString()` method can be used, the `WriteToFile()` method must be called.

ExtractString()

```
HRESULT ExtractString(BSTR strVal, VARIANT* pFieldName, VARIANT*
pFieldValue)
```

This function helps to divide the string into two parts. As a separator, either the character '\021' or ASCII code '17' must be used.

Input:

BSTR strVal	the string which is meant to be separated into two parts
-------------	--

Output:

VARIANT* pFieldName	left part of the string until the separator
VARIANT* pFieldValue	right part of the string from the separator

WinExec()

```
HRESULT WinExec(BSTR strFile, BSTR strParams)
```

This function allows starting the application and passing the parameter.

Input:

BSTR strFile	file name of the application
BSTR strParams	command line parameter

Output: -

Objects

The ActiveX control `oxactive.dll` offers the following objects:

- § ASFile Object: allows access to the AddOn or the event's handoff file
- § RequestPages object: comprises all registers of an index form (collection of RequestPage objects)
- § RequestPage Object: a single data sheet
- § ActivePage object: is a RequestPage object, i.e. a link to the active data sheet

§ ASFields Object: comprises all fields of an index form (collection of all ASField objects)

§ ASField Object: a single field of a form

These objects are used to read and edit the data of the handoff file.

Example:



The value of the 'Jahr' field (year) can be read with one of the two following methods:

```
ActivePage.ASFields.Item("Year").Value
```

```
RequestPages.Item("Workflow log").ASFields.Item("Year").Value
```

Method a) will read the field through the ActivePage object; method b) through the RequestPages collection.

ASFile Object

The ASFile object allows for access to the AddOn or the event's handoff file. This handoff file contains information on the index form.

Properties:

L – read access only

S – write access only

L/S – read and write access

ActivePage	provides a RequestPage object of the currently active data sheet	L
AddonFields	for AddOns only Provides a collection of ASField objects which are connected to the AddOn. The first element is exactly the field to which the AddOn button is assigned. The other elements of the collection allow accessing all fields which are connected with the AddOn.	L
ErrorMessage	In connection with the 'ASField.IsErrorField' property, an error text for both events 'BeforeValidate' and 'OnItemClick' can be specified, which will then be shown in an info window. A prerequisite is that the ResultCode is not	L/S

	0 (see Example 4).	
EventAction	Only for events. Available values are: NEW – a new object will be created UPDATE – a given object will be modified REQUEST – a search will be performed READONLY – a given object is open in read-only mode.	L
EventCode	Returns the numerical EventCode (see 'Client-Side Events').	L
Filename	name of the file which is meant to be used including the full path, usually the file name of the corresponding handoff file	L/S
FOLDERID	returns the ID of the folder in which the object is located	L
FOLDERTYPE	returns the numeric folder type	L
Handle	handle of the index form	L
IsErrorField	specifies whether or not this field is flagged as invalid on the index form	L/S
REGISTERID	returns the ID of the register in which the object is located	L
REGISTERTYPE	returns the numeric register type	L
RequestPages	Collection of RequestPage objects of all data sheets on the current index form.	L
ResultCode	Only for events. All changes will apply if the value is set to '1'. Default value: '0'	L/S
TARGETMAINTYPE	reads and writes the main document type.	L/S

Methods:

WriteToFile() Returns all objects and properties to the handoff file. The file name is provided by the `Filename` property.
Return values:
'10' – file does not exist
'11' – no file name specified

RequestPages Object

This object gives access to a collection of RequestPage objects.

Properties:

L – read access only

S – write access only

L/S – read and write access

Count number of elements in this collection L

Methods:

Item(VARIANT
Item)

Provides an element of this collection

Parameter:

Item – either the index or the name of the intended element can be specified here.

RequestPage Object

This object gives access to a single data sheet of the index form.

Properties:

L – read access only

S – write access only

L/S – read and write access

Active	'1' if this page is on top, otherwise '0'	L
ASFields	ASField collection of all fields on this page	L
FileCount	Number of files that belong to a document	L
ID	ID of the object	L
Name	name of the data sheet	L
ObjectType	Object type of this page	L

ASFields Object

This object represents a collection of ASField objects.

Properties:

L – read access only

S – write access only

L/S – read and write access

Count number of elements in this collection L

Methods:

Item(VARIANT
Item)

provides an element of this collection

Parameter:

Item – either the index or the name of the intended element can be specified here.

ASField Object

This object gives access to a single field of a search form.

Properties:

L – read access only

S – write access only

L/S – read and write access

CtrlPages	Collection of all registers of a page control.	L
DBName	Name of the field in the database	L
Enabled	if set to FALSE, the respective field on the form is write protected	L/S
GUID	returns the GUID of the respective field on the form	L
InternalName	the internal name of the respective field on the form	L
IsErrorField	specifies whether this field of the form is flagged as invalid (yellow) on the index form ('TRUE') or not ('FALSE') (see Example 4)	L/S
Length	Max. length of the field	L
Name	name of the field	L
Type	data type of the field (see below)	L
Value	value of the field	L/S
Visible	if set to FALSE, the respective field on the form is hidden	L/S

Available data types:

Type	Description	Database
#	Decimal numbers	DECIMAL
0	Check boxes	SHORT
1	Radio button	SHORT
9	Numbers	INTEGER
A	all characters without numbers	CHAR
C	Page control	CHAR
D	Date	DATE
G	Uppercase letters	CHAR
I	Full text index	INTEGER
L	all characters	CHAR
M	all characters	CHAR
P	Patient type	CHAR
Q	yes/no	CHAR
S	male/female	CHAR
T	Left/right	CHAR
W	Table element	CHAR
X	all characters	CHAR
Z	only numbers	CHAR

Examples of Use in VBScript

If, in enaio® client, the aforementioned objects are used through an event or the VB Script add-on, neither the ASFile object nor the CoxHelp object need to be created explicitly. Furthermore, the 'Filename' property of the ASFile object will be set automatically.

Example 1

Output the names of all data sheets:

```
for b = 0 to RequestPages.Count-1
    MsgBox(RequestPages.Item(b).Name)
Next
```

Example 2

Output all field names of the currently active data sheet as well as change the content of a field for an event:

```
for b = 0 to ActivePage.ASFields.Count-1
    MsgBox(ActivePage.ASFields.Item(b).Name)
Next
ActivePage.ASFields.Item(1).Value = 'new value'
ResultCode = 1
WriteToFile()
```

Example 3

Output the name of the AddOn field as well as its content in the data sheet and then change the content of the AddOn field:

```
MsgBox (AddonFields.Item(0).Name + ' = ' +
AddonFields.Item(0).Value)
AddonFields.Item(0).Value = 'new value'
WriteToFile()
```

Example 4

Highlight a data sheet's field on a page control in color and activate the respective page and output an error message in the form of a tooltip for the 'OnClickItem' or 'BeforeValidate' event:

```
ActivePage.ASFields("PageCtrl").CtrlPages(2).Fields.Item(2).IsErrorField = true
ErrorMessage = "Invalid entry, please correct."
ResultCode = -1
WriteToFile()
```

Example 5

Distinguish whether a query has been performed in expert mode or not by reading the #OSEXP# key of the handoff file:

```

oxhelp.GetProfString "PAGE00", "#OSEXP#", -1, rvalue, osfile
if ( rvalue = "1") then
  'Code for expert mode
else
  'Code for non-expert mode
end If

```

Server-Side Events

Server-side events are scripts which are assigned to server jobs. An assigned VB script can be executed before and after a server job is performed.

There are two types of server events which can be integrated:

§ KernelDrivenEvents (KDE (KDE)

KernelDrivenEvents can be called before or after a job is executed. Before and after job execution, all input and return parameters can be changed by the script, respectively.

§ JobDrivenEvents (JDE)

By contrast, JDEs allow that the event can to a great extent use the job's business logic without the need to implement the business logic anew.

The first implemented JDE, which was delivered together with enaio®, is part of the DoArchive job (archiving of object types). In the context of object-related scenarios, KDEs are of no use as the job iterates all documents to be archived of an object type in an internal loop. If the KDE is in spite of that executed, it would have to implement this logic (only the 'Object type' parameter would be passed) and several validity checks by itself.

The JDE in the DoArchive job can be used, for example, to additionally archive the rendition of a document which so far has been available in the WORK area.

For the purpose of a legally compliant and audit-proof long-term archive, this event will be available every time less suited file formats (.doc, .xls) – amongst others when utilizing DMS features – are meant to be saved as PDF files in the archive.

Server events, such as object and application events, are created through the 'Object search' area of enaio® client.

To integrate KDEs, select the **Common events** item in the 'Object search' area and click on the **Add event** item in the context menu. Choose the job, which is designated to have its input parameters edited by a script, from the **KernelBeforeJob** job list and the job, which is designated to have its return parameters edited, from the **KernelAfterJob** job list. The script is created or imported by using the editor window.

To integrate JDEs, select an object type in the 'Object search' area and click on the **Add event** item in the context menu. Choose the intended job from the job list in the **Server events** area.

The job list is grouped according to the following namespaces:

Namespace	Description
abn	functions used to set up and control subscriptions (notification if document inventory was changed)
adm	administrative jobs (functions used to administer system files and a few configuration tasks at the server)
ado	Database access via ADO
cnv	conversion of image files
dms	Jobs used to query and edit index data, DMS objects, relations, and portfolios while taking account of the security system
dtr	Server-side execution of the data transfer server
krn	kernel jobs (functions related to administration, licensing, session management, engine administration, and internal control)
mng	Jobs used to administer groups and users of enaio®
std	DMS jobs and archiving (WORK, cache, file, and archive administration)
tst	test executor
vtx	processing full text queries of enaio® client
wfm	Processing and managing workflow processes and models
The documentation on server jobs will be supplied upon request.	

The following JobDrivenEvents are available additionally:

§ OnSessionLogin

This event may be used to log login attempts.

§ OnObjectHistoryEntry

This event may be used to generally monitor all relevant changes made to objects.

Server Events for the Archiving Process

In terms of audit-proof archiving, the following events are available for server events:

§ BeforeOpenMedia

During an archiving process, before writing to a medium.

The name and the number of the medium will be returned.

§ BeforeStartArchiveBatch

After the number of documents to be archived was calculated but before the archiving process begins.

The name and the number of the object type as well as the number of documents to be archived will be returned.

§ OnCloseMedia

After the archiving on a medium within an archiving process was completed.

The name and the number of the medium will be returned.

§ OnFinishArchiveBatch

After an archiving process was completed.

The name and the number of the object type as well as the number of successfully/defectively archived documents and statistical data from the report file will be returned.

§ OnFinishMedia

After a medium was finally finished. That is in case of errors, if there is no more free space or if blocked space is the only storage space still free.

The name and the number of the medium will be returned.

According to the context, the return values 'BreakJob' (cancel job) and 'BreakMedium' (do not use medium) may be available.

Server-Side Scripts

Scripts which are run on the server, may need to access the server's kernel objects. For this purpose, the server provides the 'running context' (RC) object in the script. If not defined otherwise by the JobDrivenEvents, the variable instance reads 'RC'. The different jobs in JobDrivenEvents may furthermore provide proprietary objects. These will separately appear in the documentation of the respective job. The kind of how these objects are provided depends on the job.

Script Development

All kernel objects are assigned to the running context and, opposed to objects which are added by the job itself, will never be seen directly in the global namespace of the script. These jobs may be visible in the global namespace. Anyhow, the kernel objects are additionally assigned to the running context. Kernel objects thus differ from job-specific objects because kernel objects will only be visible through the running context and never directly, whereas job-specific objects will always be visible through the running context and maybe directly as variables. The running context additionally allows accessing job-specific objects.

There are two ways to access the objects through the running context thanks to the 'Item' property, for example either 'RC.Item("SessionData") or 'RC.SessionData' (separated from the RC).

The first way accepts names and numbers. The object number can be passed as a parameter to the 'Item' property: RC.Item(2).

The 'RC.Count' property permits to determine the total number of objects in the RC. Kernel objects will always be listed behind all job-specific objects, i.e. the numbering changes for each call.

RC

In addition to all objects, the running context has the 'GUIAvailable' property which can be used to determine from within the script whether the script allows GUI calls and the 'NewJobsParams' method; the latter may be used to create a new instance of the parameter list due to which jobs can be run.

The third method of the RC is 'IncludeFile(BSTR FileName)' which is used to separate large scripts into multiple scripts.

Another RC property is 'UseNewDB'. It is used to control whether jobs can be called in a transaction or whether they receive an extra transaction context.

Objects

The following kernel objects of the first level are available:

- § SessionData,
- § ServerData,
- § General,
- § Logger,
- § Jobs,
- § Actions,
- § Registry.

Each object will be described in the following.

SessionData

This object provides information about the user account in the context of which the script will be run. The following properties are currently available:

Name	Description
SessGUID	GUID of the client session
UserName	User name
StatName	Name of the enaio® client computer
InstName	Name of the program to which the user has logged on
UserGUID	GUID of the user
StatGUID	GUID of the enaio® client station This is not a MAC address but the GUID of the data record in the database.
UserID	User ID
Supervisor	The supervisor flag from the user table of the user
LangID	Language ID from the user table of the user

ServerData

This object provides different configuration data from the runtime environment of the server. The following properties are currently available:

Name	Description
DataDir	Path to the data area of the server group
RootDir	Path through which the server was started
TempDir	Path to the ostemp directory of the server
ConfDir	Path to the etc directory of the server group
UserDir	Path to the user directory of the server group
Service	Name of the service
LogDir	Path to the log directory into which the server saves its logs
Connect	Computer name and TCP port, separated by the hash character '#'
ServerID	ID of the server
GroupID	ID of the server group

General

This object provides some general information on the job. The following properties are currently available:

Name	Description
JobNumber	Number of the job, sequentially numbered since server start
ThreadID	ID of the thread in which the job is currently performed

Logger

This object allows for logging within the kernel context. The script may also keep its own log by directly addressing the `oxrpt.dll` library via COM. In doing so, an individual configuration is used. The script may alternatively log through the logger object. The logger object provides the following methods:

Name	Description
Flow	Flow log on level 5
FlowEx	Level and script location can be defined
Info	Flow log with facility 'informational' on level 3
InfoEx	Like info, the script location can be defined
Warning	Flow log with facility 'warning' on level 3
WarningEx	Like warning, the script location can be defined
Error	Error log with facility 'error' on level 0
errorEx	Like error, the script location can be defined

The call of a log entry will appear in the log as 'SCRIPT: [...]' text, while the passed text is logged in the square brackets. According to the called method, the entry will appear as 'Flow', 'Info', 'Warning', or 'Error'.

All methods have a 'Text' parameter which represents the actual log entry and, dependent on the method, further parameters:

```
Flow(BSTR Text);
Warning(BSTR Text);
Info(BSTR Text);
Error(BSTR Text);
FlowEx(BSTR Text, UINT Level, BSTR Function, UINT Line);
WarningEx(BSTR Text, BSTR Function, UINT Line);
InfoEx(BSTR Text, BSTR Function, UINT Line);
ErrorEx(BSTR Text, BSTR Function, UINT Line);
```

The 'Flow' method logs on level 5, the methods 'Info' and 'Warning' use level 3, and the 'Error' method logs on level 0.

Jobs

This object allows for the execution of several server jobs. The jobs are run synchronously, i.e. the calling method will exactly be returned when the job is finished, no matter if it was successful or not. A job call may look like this:

```
1 Dim PrmIn
2 Dim PrmOut

3 Set PrmIn = RC.NewJobsParams
4 Set PrmOut = RC.NewJobsParams

5 PrmIn.Clear()
6 PrmIn.Value("Flags") = 0
7 res = RC.Jobs.krn.GetNextIndex(PrmIn,PrmOut)
8 if res = 0 then
9 MsgBox("krn.GetNextIndex succeeded: " & NextIndex)
  'On parameter determination see the next chapter.
10 else
11 MsgBox("krn.GetNextIndex failed: " & res)
12 end if
13 PrmOut.Clear()
```

In the lines 1 to 4, two objects are created which serve as input and output parameters of the job. These objects cannot only be used for one but for a number of jobs. Elements can be added and queried but not deleted separately. Therefore, the 'Clear' method is introduced as it will clear the list (line 5 to 13). The list can consequently be filled with input parameters. The property, which represents the respective parameter, is named 'Value' of which the argument is the parameter name. The type of the parameter will be analyzed, thereby creating an XMLRPC parameter with the same type. 'BSTR' will be converted to 'XMLRPC string'. Different 'Integer values' become 'XMLRPC integer' and 'BOOL' becomes 'XMLRPC boolean'. The present version does not support other 'XMLRPC types'.

Afterwards, the call is performed as follows:

```
RC.Jobs.namespace.jobname(in-list,out-list)
```

Here, 'namespace' is the name of an executor or namespace (three characters as usual) which is followed by the job name. The job call returns an integer value which usually is set back to '0' if the job was successful. This value represents 'dwResult' which returns the actual job function. If problems occur which prevent

the job from being executed (e.g. an exception or either the job or the namespace is unknown), instead of passing a return value, a COM exception will be generated. If the 'Job.krn.MyJob' call returns without triggering a COM exception, it can be assumed that MyJob has been executed; and if, for example, it returns the value '-1', this value is passed for sure by the job function and not by the kernel.

In order to pass files to the job, another parameter must be added. This parameter is called \$Job\$Files\$ and is a string. It consists of file names which are separated by semicolons.

After return, output parameters can be interpreted; in the example they are located in the 'PrmOut' list of output parameters. Naturally, the parameter names must be known. The output files are obtained by analyzing and splitting the job.

In doing so, note that a question mark may be prepended to each file name. This will be the case if the job has created temporary files in the ostemp directory, e.g. when decrypting a file from the WORK directory into the ostemp directory.

If there is no question mark prepended to the file name, it is not a temporary file and has not been filed in the CACHE or WORK directory either.

If, for example, a non-existing variable is queried, the parameter is consequently not returned; the variable has neither been initialized and has the type 'VT-EMPTY'. The 'IsEmpty' function can be used to check it in the script. Job names and names and types of job parameters are described in the server API documentation.

If BASE64 parameters are meant to be passed to the job or to be read out of the job, consult the following section on 'Parameters'. If, for example, the DOM variable is an MSXML.DOMDocument object into which an XML was loaded, the command

```
PrmIn.Value("Buffer") = Dom
```

is used to assign an input parameter of the type BASE64 to the job. Vice versa, an output parameter can be read out as follows:

```
B64Result = PrmOut.Value("Result")
```

```
Dom.load(B64Result)
```

When calling a job at the server and passing the \$\$\$Server ID\$\$\$ parameter to it, this job is forwarded to the transferring server and executed. For security reasons, this process is only available if the job is called in a script at the server via RC.

Actions

This object allows for the execution of functions which are implemented by the kernel itself. The following methods are currently available:

Name	Description
SendMail	Sends an e-mail which contains the addresses of recipient and sender, subject, text, and list of files.
SendAdminMail	Sends an e-mail to the administrator provided that his e-mail

	address has been specified in the registry.
StreamReset	Clears an IStream.

The list of file names is a semicolon-separated string. The methods have the following parameters:

```
SendMail(BSTR To, BSTR From, BSTR Subject, BSTR Text, BSTR Files);
SendAdminMail(BSTR From, BSTR Subject, BSTR Text, BSTR Files);
StreamReset (VARIANT stream); // VARIANT is of type IStream or BYREF
| VARIANT and then IStream
```

Registry

This object allows reading elements out of and writing elements to the server registry. The following properties are available:

Name	Description
Item	This property can be read and written to, and has a parameter of the 'BSTR' type which represents the full path to the element. The element's value has also the 'BSTR' type.

The path to the element begins with the current scheme. For example, the 'ComString' element, which is located under the current scheme, is accessed as follows: `RC.Registry("ComString")`.

The 'Schema' element, which is located under the 'DataBase' key, is accessed as follows: `'RC.Registry("DataBase\Schema")`. Take account of the case-sensitivity of the entry.

Parameters

Several parameters can be passed to the script. Likewise, the parameters can be returned after the script's execution.

```

Dim InputNames
InputNames = RC.InputParams.Names
MsgBox("Input params: " & InputNames)

Dim NameArray
NameArray = Split(InputNames, ";", -1, 1)
MsgBox("VarType(NameArray) = " & VarType(NameArray))

Dim sMsg, i, sName
Dim Param
for i = LBound(NameArray) to UBound(NameArray)
    sName = NameArray(i)
    if (len(sName) > 0) then
        Param = RC.InputParams.Value(sName)
        if (IsEmpty(Param)) then
            sMsg = sMsg & "Parameter " & sName & " not found"
        else
            sMsg = sMsg & sName
            sMsg = sMsg & ": "
            sMsg = sMsg & Param
        end If
        sMsg = sMsg & vbCrLf
    end If
next
MsgBox(sMsg)

RC.OutputParams.Value("hallo") = "Welt"

```

Both objects 'RC.InputParams' and 'RC.OutputParams' enable working with parameters. The 'Names' property returns the names of all listed parameters, separating them by semicolon. The 'Value' property enables reading and writing the parameter.

A particular case of script execution is the execution of scripts of KernelDrivenEvents. The KDE's will be run before and after the job. Therefore, they must access the job parameters. This is ensured due to the variables 'InputParams' and 'OutputParams'. In the Before script of a KDE, all parameters are provided to the script as input parameters. After the script's execution, this list of input parameters will entirely replace the parameter list of the original job because single parameters can be removed from the list, as well. The After script of a KDE provides all input parameters of the job (or the Before script) as well as all output parameters. Afterwards, the output parameters can be manipulated. In contrast to the Before script, output parameters will not be entirely replaced but the values of available parameters will be modified.

The values of parameters can be queried and edited as described above, except for parameters of the 'BASE64' type. The latter require another procedure as they are represented as IStreams. The script can pass IStreams only to those interfaces that are capable of identifying IStreams, for example an XMLDOMDocument because the 'msxml' implementation allows loading XML from IStreams. A DOM may subsequently be again saved to an IStream. However, the IStream must be cleared first as the content of the 'DOMDocument' will be added to it. The 'RC.Actions.StreamReset(param)' can be used for this purpose.

Files

The running context provides methods which allow passing file lists to the script and getting them back.

The 'RC.InputFiles' can be used to read out the file list from within the script which the script has received as an input list.

The 'RC.OutputFiles' can be used to define the output list of files in the script.

Example: `RC.OutputFiles = "c:\temp\1.txt;c:\temp\2.txt"`

File names must be separated by semicolon.

RunScript

The 'krn.RunScript' job allows testing the described functionality. It can be additionally used for administrative purposes, such as the frequent execution of particular scripts.

The 'RunScript' job requires the following parameters:

Name	Type	Description
Flags	Integer (2)	A value of '1' will not remove the files which were passed to the job.
Script	String (1)	Text of the script. Provided that the parameter is an empty string, it will be checked whether at least one file is received with the job. If not, an error will be returned. If yes, the file content will be used as the script text. Subsequent to the job, all incoming files will be deleted provided that 'Flags' has a value of '0'.
GUI	Boolean (3)	Will be forwarded as 'bGUIAvailable' to the performer. A value of '0' will deactivate the option to show a message box. Vice versa, a value of '1' activates the option unless it has been generally disabled in the registry.
CtxName	String (1)	Name of the running context under which it is made visible in the script. If this parameter is empty, the name 'RC' will be used.
Main	String (1)	Name of the function to be carried out by the script; may be empty as well. If the parameters is missing, 'Main' will be used as the function.
Eval	Boolean (3)	Specifies whether or not the script text represents an expression.

After job execution, the return value of the script function (the function is called 'Main' if the 'Main' job parameter is empty) which was executed on the server's side will be returned as '\$ScriptResult\$' return value (of the data type string).

Example:

```

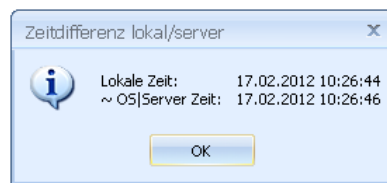
Set oServer = CreateObject( "OxSvrSpt.Server" )
set oSession = oServer.Login("<User>" , "<PWD>" , "<Server>",
"<Port>" )

sScript= _
"Function Main" & vbCrLf & _
"Main=now" & vbCrLf & _
"End Function"
MsgBox _
"Local time: " & vbTab & now & vbCrLf & _
"~server time: " & vbTab & _
RunScript(oSession, sScript), vbInformation, _
"Time difference local/server"

Function RunScript(oSession, sScript)
    set oJob = oSession.NewJob("krn.RunScript")
    oJob.InputParameters.AddNewStringParameter "Script", sScript
    oJob.InputParameters.AddNewStringParameter "CtxName", ""
    oJob.InputParameters.AddNewIntegerParameter "Flags", 0
    oJob.InputParameters.AddNewBooleanParameter "GUI", 0
    oJob.InputParameters.AddNewBooleanParameter "Eval", 0
    oJob.Execute
    RunScript = oJob.OutputParameters("$ScriptResult$").Value
End Function

```

The following message box will be returned as the result:



To pass a parameter list to the script, further optional parameters can be passed to the job. These parameters must differ from normal job parameters by having other names than available for this job. What is more, the names and values must be separated as follows upon execution:

Name	Type	Description
\$SP_1_name\$	String (1)	name of the first script parameter
\$SP_1_value\$	Random	value of the first script parameter
...		
\$SP_xx_name\$	String (1)	name of the last script parameter
\$SP_xx_value\$	Random	value of the last script parameter

These parameters have to be numbered sequentially.

Global Scripts

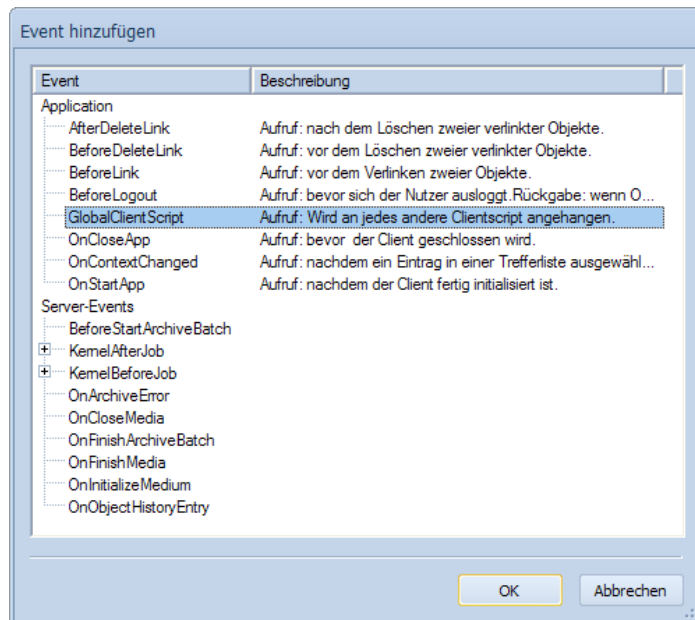
Global scripts are used to provide constants, subprograms and functions. It is possible to set up both a global client script and a global server script.

Before execution, data of the global client script will always be added to the scripts of client events, whereas data of the global server script will always be added to the script of server events.

A global object type script can also be created for each object type. Data are stored there which are only used for this object type. You can set up object type scripts using the context menu of an object type in the 'Object search' area.

Due to this organization, maintenance and administration of script code is significantly simplified.

Select the **Common events** item in the workspace and click on the **Add event item** in the context menu.



Either select the **GlobalClientScript** or the **GlobalServerScript** and confirm with **OK**.

The editor window will open. The script can be edited.

The executable script code which may be contained in a global script will be carried out at every corresponding event. If the result of run script code is 'false', a still pending job will not be executed.

Example of event script code related to a GlobalClientScript:

```
MsgBox TextForString
useGlobalScript

Sub useGlobalScript
    Dim ret
    call globalHello
    ret = Dummy("String from calling Fct")
    MsgBox
End Sub
```

Example of the corresponding GlobalClientScript:

```

const TextString = "String from global script"
const TextForDummy = "Function successfully executed Input="

Sub globalHello()
    MsgBox"Hello from global script"
End Sub

Function Dummy (text)
    Dummy = TextForDummy + text
End Function

```

Controlling the Info Window

Next to the 'Object search' area and the search bar, a third window is available in enaio® client: the info window. This window cannot be designed by the user but only administratively with events.

An URL or an HTML string is passed to the info window. It has a COM interface which can be addressed under the name 'InfoWindow' from within every event script.

The following properties and methods are implemented in the interface:

Property	Description
ID	unique ID of the window
Visible	specifies whether the window is visible
Caption	text in the title bar of the window
URL	indicates the URL which is currently shown or to be shown
Closeable	specifies if the window can be closed by the user
EnableContextMenu	specifies whether the context menu of the Internet Explorer is meant to be shown
HtmlDocument	specifies the current HTML document

Method	Description
ShowHtml(String html)	displays the handed over HTML string in the window
Refresh()	updates the view
GoBack()	corresponds to the 'Back' button in the Internet Explorer
GoForward()	corresponds to the 'Forward' button in the Internet Explorer

Event Administration

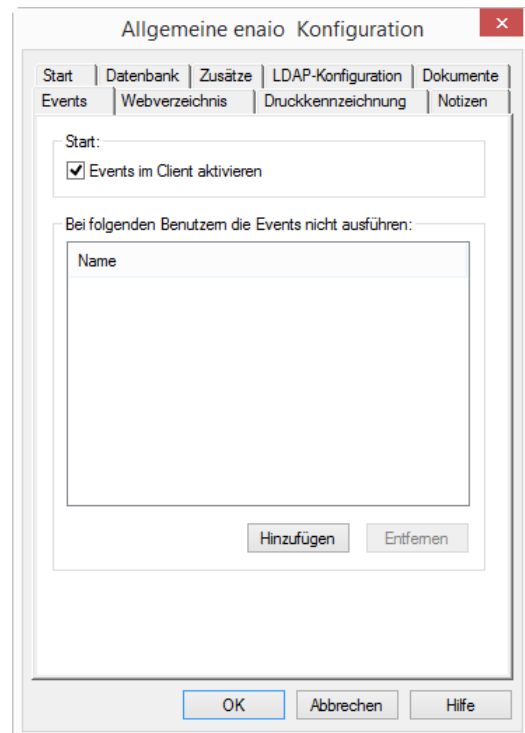
You can set whether and for which users events are executed in enaio® client. In enaio® webclient, events are always executed for all users.

To do so, open the **General enaio configuration** dialog box by clicking the **Entire System** button in the ribbon.

If, on the **Events** tab, you select the option **Enable events in client**, events will be executed for all users who are not entered in the user list.

If you disable this option, the events are only executed for the users that are entered in the user list.

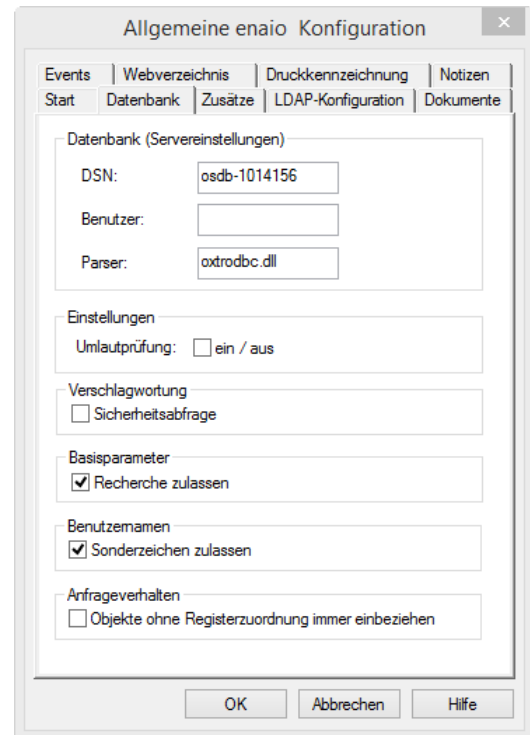
By clicking the **Add** button the user list can be created.



After having modified events, other users must restart enaio® client or update the events by pressing the keyboard shortcut **Shift+Ctrl+F5** in order to apply all changes.

The query behavior of integrated scripts must be configured with enaio® administrator.

When activating the option **Always include objects without register assignment** on the **Entire system/Database** tab, documents without register assignment, i.e. which are not located in a register, are also exported, whenever register and document data are queried. This setting will not have any effect as long as the query behavior is specified in the script itself.



For searches in enaio® client, users can configure the query behavior in the 'Query behavior' area of their user-specific personal settings dialog.

enaio® editor-for-events

enaio® editor-for-events – Introduction

enaio® editor-for-events is used to assign a script to a DMS object and an event and to save the event in the database.

enaio® editor-for-events is an integral part of enaio® client. Given that a user is provided with all necessary system roles and licenses, the corresponding functions are activated in enaio® client.

To test events, enable the debug mode of enaio® client (see 'Debugging').

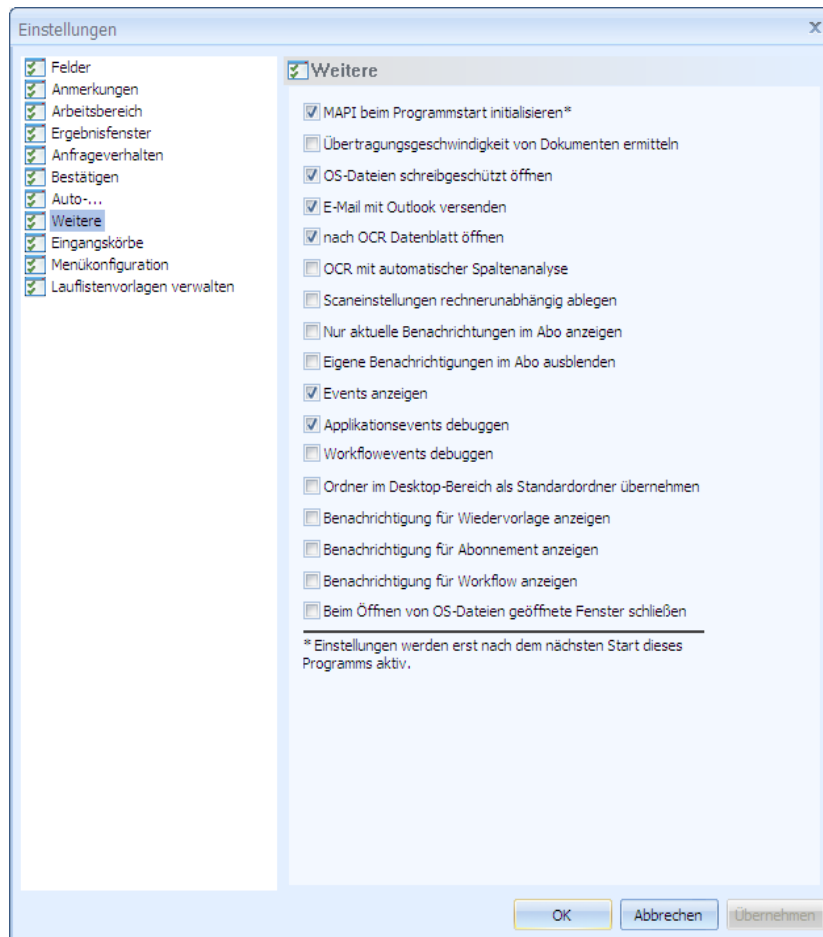
Follow these steps to create events:

- § Select a DMS object or enaio® client from the 'Object search' area,
- § Select an event,
- § Insert the script code,
- § And save the event to the database.

The event will be subsequently executed for all administrated users (see Event Administration') who restart enaio® client or update the security system by pressing the keyboard shortcut **Shift+Ctrl+F5**.

Creating Events

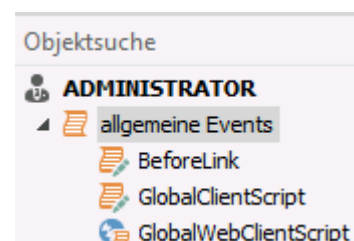
Access to these functions can be configured in the **Settings** area.



Enable the option **Show events** in the **More** section to access the functions in the workspace of enaio® client. Set up events will be displayed there.

At the top in the 'Object search' area, the **Common events** entry will be shown. All events which are triggered by logging in and out of enaio® client or starting and exiting the client will be listed.


Other events are triggered by actions related to DMS objects and are thus assigned to these object types within the tree structure.



 Server-side events

 Client-side events

 Webclient-side event

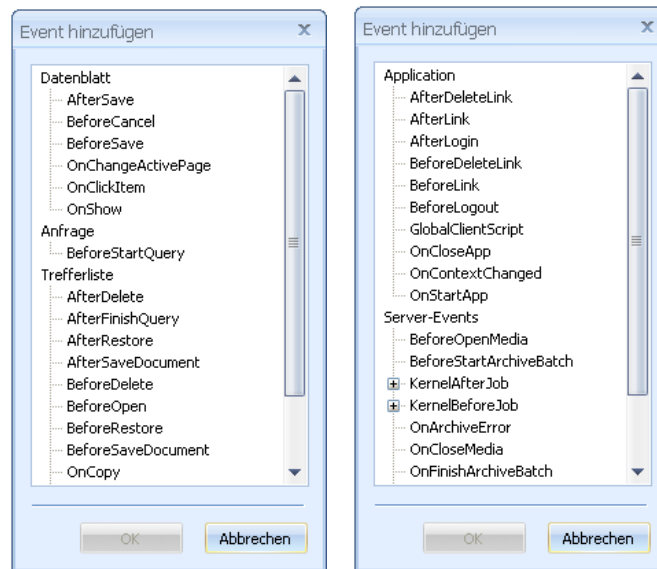
 Client-side change events in batch mode

How to create an event:

§ Select the **Common events** entry or a DMS object.

§ Select the **Add event** entry from the context menu.

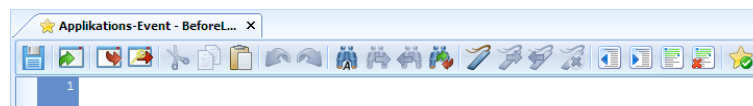
The **Add event** dialog will open.



It will list all those events which can be assigned, either object-related, application-related or server events.

§ Select the desired event by double-clicking on it.

The editor window will open.



§ Enter the VB script into the editor window.

§ Click on the  **Save script** button.

The event will be stored in the database.

Each event can only be assigned to an object once.

If you use scripts to refer to dialog elements containing special characters, errors may occur. In this case, use internal names for referring to dialog elements.

Importing Scripts

The events written by OPTIMAL SYSTEMS will be provided as files in encrypted format which can be imported, either assigned to a DMS object or an application, and saved to the database.

It is also possible to import and save single events as encrypted files in enaio® editor-for-events.

Unencrypted files cannot be imported.

How to import an event:

§ Select the **Common events** entry or a DMS object.


§ Select the **Import event** entry from the context menu.

§ Select the file using the file selection dialog.

Encrypted event files have the file extension 'evc'.

The event will be shown in the workspace and the editor window will open showing the VB script.

User will be notified if the name of the assigned DMS object does not correspond to the name of the DMS object in the event file.

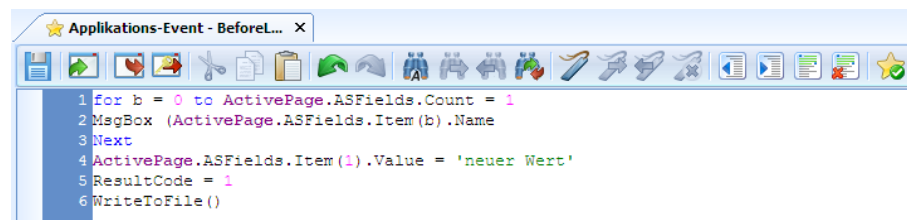
§ Click the  **Save script** button in the editor window.

The event will be stored in the database.









The Editor Window















When creating an event, the VB script is inserted into and edited in the editor window. When importing an event, the VB script is displayed in the editor window in unencrypted format.

When opening an event from the database, date, time and name of the user who has most recently modified the script will be shown in the status bar.



The toolbar of the editor window offers the following buttons:

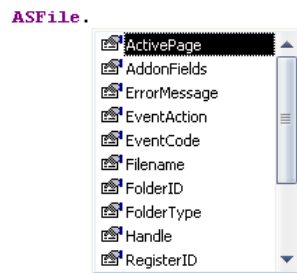
-  **Save script**
Saves the script to the database (CTRL-S).
-  **Import script**
Imports a script from a file.
-  **Export script**
Exports the script to a file.
-  **Encoded export**
Exports the current script in encoded form to a file.
-  **Cut**
Cuts out the selected text.
-  **Copy**
Copies the selected text into the clipboard.
-  **Paste**
Pastes the text from the clipboard into the current cursor position.
-  **Undo**
Undoes the last action.

-  **Restore**
Restores the previous version after the action has been undone.
-  **Find**
Enters a search term and a search direction.
-  **Down**
Find the next result of the searched expression in the text.
-  **Up**
Find the previous result of the searched expression in the text.
-  **Replace**
Enter a search term and a term that the search term will be replaced with.
-  **Bookmark on/off**
Adds a bookmark to the currently selected line or removes it.
-  **Next bookmark**
Switches to the next bookmark.
-  **Previous bookmark**
Switches to the previous bookmark.
-  **Delete all bookmarks**
Deletes all bookmarks.
-  **Decrease indent**
Decreases the indent of the selected lines.
-  **Increase indent**
Increases the indent of the selected lines.
-  **Comment out block**
Comments out the selected lines.
-  **Remove comments for block**
Removes comments for the selected lines.
-  **Syntax check**
Starts the syntax check.

Most of these features are also available from the context menu.

With **Ctrl+G** you can specify the line number you want to go to.

enaio® editor-for-events supports Intellisense: having entered an object name and pressed the period key, a context menu will open showing all methods and properties of the object:



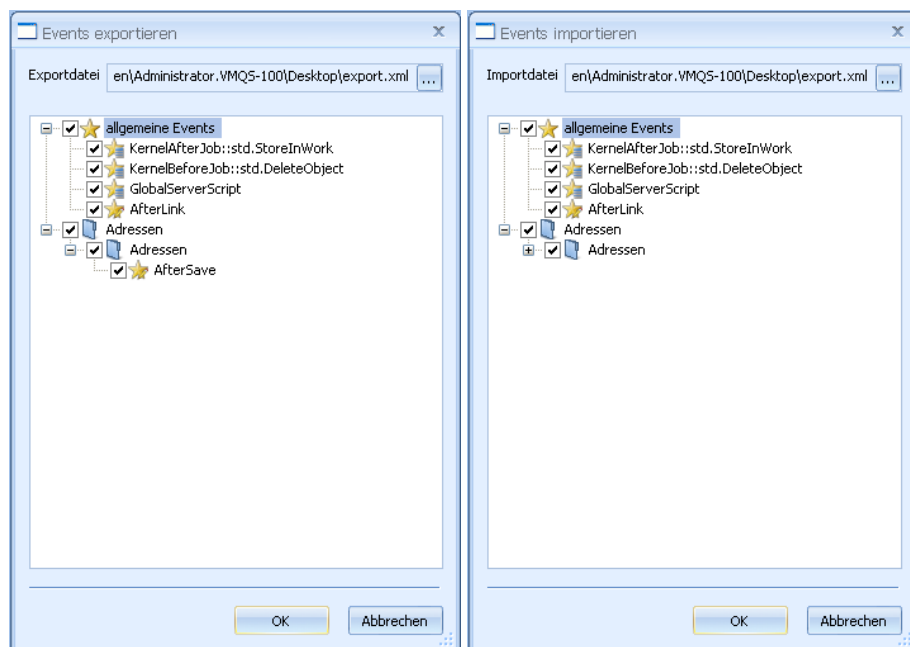
Export/Import

The editor window may be used to import and export single scripts (as well in encrypted form).

The 'Object search' area of enaio® client can be used to export and import events.

Script files exported this way are given the file extension *.evc.

When selecting a user name in the 'Object search' area, the context menu will offer both **Event export** and **Event import** functions.



The event export creates a number of files: the assignments of object type and script are saved in an XML file. The actual scripts are encrypted and individually exported as *.evc files into the same directory.

Select the events to be exported and confirm the selection by clicking **OK**.

For the event import select the XML file containing the assignments and specify which events are meant to be imported. The scripts must be located in the same directory as the XML file.

Users will be notified if scripts have already been assigned to the object types and can decide if they want to overwrite them.

Debugging

To test events, activate the debug mode in enaio® client. Events are not executed in debug mode; instead the event editor will open the script of the respective activity. The script can be then executed step by step, whereas current variable values will be displayed and values can be changed.

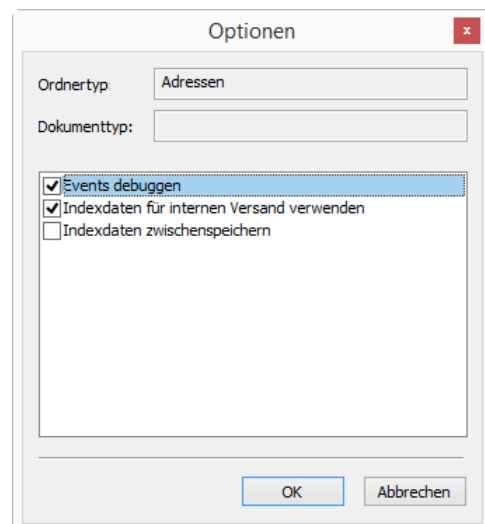
The script cannot be edited in debug mode.

In addition to application-related and object-related events, scripts, which are integrated in a workflow process, can be opened in the event editor and performed step by step.

The debug mode for application-related events is activated in the **Settings** area. To do so, activate the option **Debug application events** in the **More** section.

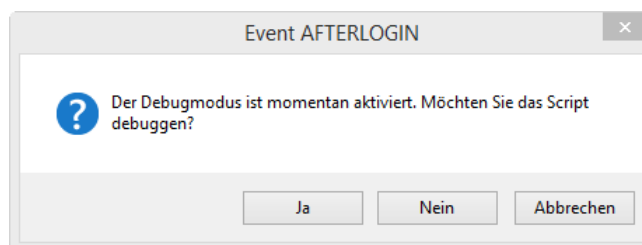
The debug mode for single object types is enabled in the Options dialog which is opened through the context menu.

Select the **Debug events** item.



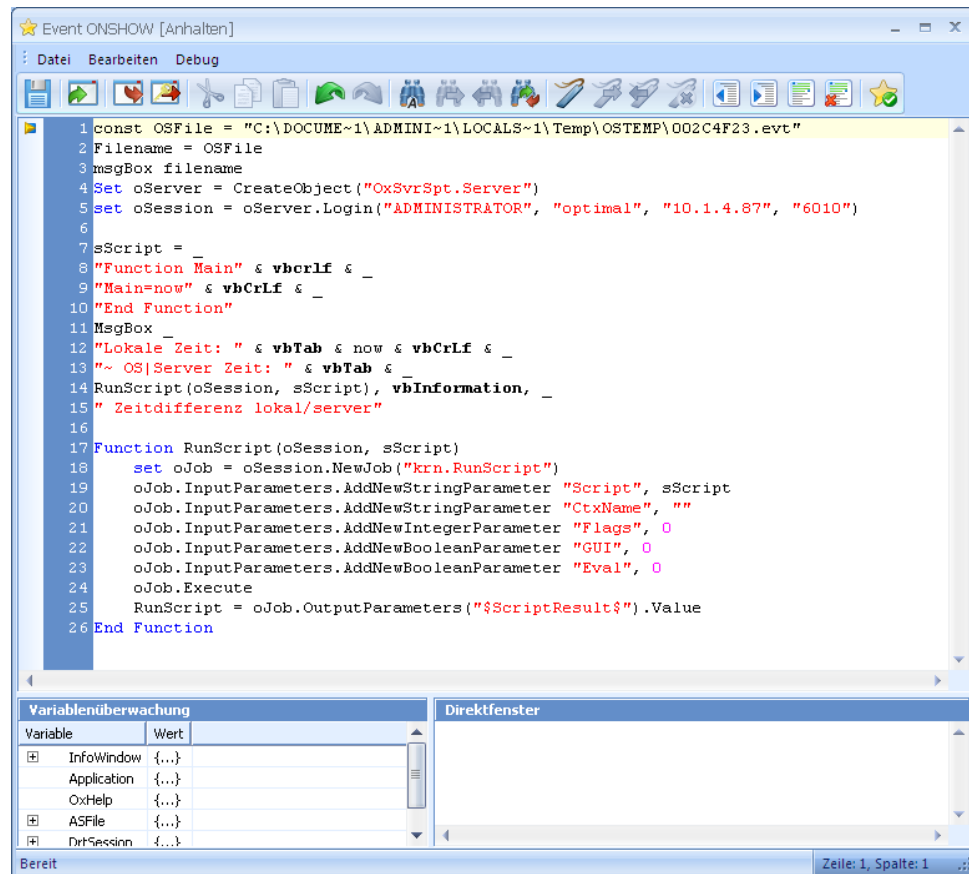
To test workflow scripts in debug mode, open the **Settings** area and activate the option **Debug workflow events** in the **More** section.

Provided that the debug mode is active, a confirmation dialog will open once a respective event is triggered.



Click **Yes** to confirm the debug mode, click **No** to let the event be executed or click **Cancel** to continue without the event.

When confirming the debug mode, the script will open.









In debug mode, the window is divided into three areas: the script, **Variable monitoring**, and **Direct window** areas.

The **Variable monitoring** area provides all variables, objects, and values which are available in the current script. The data cannot be edited.

The **Direct window** can be used to check expressions and change variable values:

- § Enter an expression and confirm with **Enter** to verify the expression.
- § Assign a value to the variable and confirm this with **Shift+Enter** to change the value of the variable.

Test the event script with the following buttons:

-  **F9** Breakpoint on/off
-  **F5** Start debugging until the next breakpoint or until the end
-  **Shift F5** Exit debugging
-  **F8** Jump to call or function
-  **Shift F8** Execute next command in one step
-  **Ctrl Shift F8** Step back, jump out of the function

Press the key combination **Ctrl+B** to open the **Breakpoints** dialog in which all defined breakpoints will be listed.

Index

A

Administration 65
AfterDelete 36
AfterDeleteLink 39
AfterLink 39
AfterRestore 40
AfterSave 30
AfterSaveDocument 40
ASField object 49
ASFields object 49
ASFile object 47

B

BeforeCancel 40
BeforeDelete 34
BeforeDeleteLink 39
BeforeLink 38
BeforeOpen 33
BeforeRestore 40
BeforeSaveDocument 39
BeforeStartQuery 32
BeforeUndoCheckOut 35
BeforeValidate 27

C

changes 14

D

debug mode 73
debugging 73
dialog element 'Tables' 44
direct window 74

E

editor window 70
enaio® webclient 5
encryption 69

H

Handoff files 15

I

importing 69

installation 5

J

JobDrivenEvents 52

K

KernelDrivenEvents 52

L

license 5

M

methods 45

O

OnAddLocation 37
OnChangeActivePage 41
OnClickItem 21
OnCreateCopy 38
OnMove 36
OnMoveExtern 37
OnShow 24
oxactive.dll 16, 45

P

PDF file 4

Q

Quickfinder 27

R

RequestPage object 49
RequestPages object 48
return values 16

S

StartAction 15
system role 5

T

Temporary directory 15

V

variable monitoring 74

W

write-protection of fields 27