

# Software Documentation

## enaio® client – Programming Reference

Version 9.10

All software products and all related programs and functions are registered and/or used trademarks of OPTIMAL SYSTEMS GmbH, Berlin or one of its companies. They may only be used with a valid license agreement. The software and its associated 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 or screenshots are fictional. Any resemblance to existing companies or persons is purely coincidental and unintentional.

Copyright 1992 – 2019 by     OPTIMAL SYSTEMS GmbH  
  Cicerostraße 26  
  D-10709 Berlin

03.12.2019  
Version 9.10

# Contents

Introduction .....	6
Handoff Files .....	6
Internal Names .....	6
Logical Expressions and Expressions .....	7
Clauses .....	7
Expressions .....	7
Wildcard in Query Files .....	10
Code Examples .....	11
VB and VBA .....	11
Object Main Types .....	12
All COM Commands .....	13
ActivateApp .....	13
AdjustRetention .....	14
AppBrowserSession .....	15
ApplicationLogin .....	16
CalcFileDigest .....	17
CheckInDocument .....	18
CheckLicence .....	19
CheckObjectAccess .....	20
CheckOutDocument .....	21
ClearSignatureProperties .....	22
CloseAllWindows .....	23
CloseObjectID .....	24
ConvertImage .....	25
CopyObject .....	26
CreateDocumentLink .....	27
CreateMimeFile .....	28
DecodeIMAPFile .....	29
CreateNewDocShare .....	30
DeleteFromArchive .....	31
DoPrefetch .....	32
ExecuteRequest .....	33
ExecuteRequestEx .....	34
FindObjectType .....	35
FindObjectTypeEx .....	36
FreeDocument .....	37
FreeDocumentEx .....	38
GenerateColdFiles .....	39
GenerateOSFile .....	40
GetActiveDocument .....	41
GetAllArchives .....	42
GetAllOpenFolders .....	43
GetCurrentResultList .....	44
GetCurrentSelection .....	45
GetDataID .....	46
GetDescription .....	47
GetDocTypesFromArchive .....	48
GetEnvironment .....	49
GetFilesFromID .....	50

GetFilesFromIDEx .....	51
GetImageType .....	52
GetLastError .....	53
GetMainType.....	54
GetObjectFields .....	55
GetObjectFieldsEx.....	57
GetObjectName.....	59
GetObjectNameEx .....	60
GetObjectPath .....	61
GetObjectPathEx.....	62
GetObjectPattern.....	63
GetObjectTypeInfoImage .....	64
GetRegTypeFromArchive .....	65
GetResultFields.....	66
GetSignatureProperty .....	67
GetSelectedObject .....	68
GetSignDocumentResult .....	69
GetWDocPattern.....	70
GetWDocPatternNames .....	71
GoToDocPage .....	72
InfoWindow .....	73
InsertFileList.....	74
InsertFileListS.....	76
InsertIntoArchive .....	77
InsertIntoArchiveS.....	78
InsertIntoDocument .....	79
InsertIntoDocumentS .....	81
InsertIntoRegister.....	82
InsertIntoRegisterS.....	83
InsertNewDMSObject.....	83
LicLogin .....	84
LicLogout.....	85
LinkDocuments.....	86
MergeArchives.....	87
MoveObject .....	88
OleDdeRequest.....	89
OpenAboDialog.....	90
OpenDataDlg.....	91
OpenObjectID .....	92
OpenResultList .....	93
OpenObjectIDEx.....	94
OpenURL.....	95
OpenWorkItem .....	96
PrintDocumentID .....	97
RefreshFolderWindow .....	98
ScanDocument .....	99
ShowVariantsDialog.....	100
SelectObject .....	101
SendMail.....	102
SendMailMapi .....	103
SetPlannedRetention.....	104
SetResultListSelection .....	105
SetSignatureProperty .....	106
SetWaitingCursor.....	107
SignDocument.....	108
SignDocumentEx .....	109

StartArchiveRequest.....	110
StartDocRequest.....	112
StartRegRequest .....	114
StartClientRequest.....	116
StoreNotice .....	117
TransformXML .....	118
UndoCheckOut .....	119
UnlinkDocuments.....	120
UpdateArchiveData.....	121
UpdateArchiveDataS.....	122
UpdateDocFileList.....	123
UpdateDocShare .....	124
UpdateDocumentData.....	125
UpdateRegisterData .....	127
COM interface of preview window .....	129
Introduction .....	129
All COM Commands.....	129
Script interface of preview windows.....	139
Introduction .....	139
Functions to Browse Across Documents.....	139
Functions to Control Client Windows .....	141
Functions for Dashlet Control .....	146
Appendix: Configuring the enaio® Printers .....	151
Introduction .....	151
The Configuration File.....	151
Logging .....	153
Terminal Server .....	154
Appendix: Structure Tree Catalogs .....	155
File Conversion .....	155

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## Introduction

enaio® provides a COM interface to communicate with the client. This interface exists since optimal\_AS® 3.x. In early versions of optimal\_AS® 3.x the COM interface co-existed with the DDE interface, before that communication with the client was only possible with DDE.

**COM** stands for **Component Object Model**, which is a model introduced by Microsoft to allow communication between Windows applications. Frequently, the term **OLE** is used, which stands for **Object Linking and Embedding**. In this document the term **COM** is used exclusively. In this handbook you will find source code examples, information on the transition from the DDE interface of enaio® to the COM interface, and a description of all available COM commands.

## Handoff Files

Files are passed to the COM commands for insertion and query. Files that hand over insert commands (**InsertIntoArchive**, **InsertIntoRegister**, **InsertIntoDocument**) contain the indexing of the objects to be created. Files used for request commands (**StartArchiveRequest**, **StartRegRequest**, **StartDocRequest**) contain request information i.e. the wanted indexing. Both file types additionally contain labels of the requested / to be inserted objects.

The handoff file for the **InsertIntoArchive** command may have the following structure:

```
[EINFÜGEN]
SCHRANK=Schranksname
FIELD1=field_value#1
FIELD2=field_value#2
...
FIELDn=field_value#n
```

A handoff file for the **StartArchiveRequest** command may have the following structure:

```
[ANFRAGE]
SCHRANK = cabinet name
KLAUSEL1=cabinet_name@field_name1=field_value1
KLAUSEL2= cabinet_name@field_name2=field_value2
...
...
KLAUSELn= ...
DATENFELDER=0 (1)
DATENHEADER=0 (1)
ANFRAGEFENSTER=0 (1, 2)
AUTOSTERN=0 (1,2)
```

Note: Handoff files differ from the COM commands. Please note the spelling of the COM commands.

## Internal Names

The object names contained in the handoff files can be replaced by internal names. Thus percent sign needs to be placed before and after the internal name.

Example of a query file with internal names:

```
[ANFRAGE]
SCHRANK = %internal_cabinet_name%
KLAUSEL1=cabinet_name@%internal_field_name1%=field_value1
KLAUSEL2= %internal_cabinet_name%@field_name2=field_value2
```

Examples for an expression with internal object names:

```
AUSDRUCK1=%InboundDocument%@Date1^6^10.03.2000
```

Example for a logical expression:

```
KLAUSEL1=%InboundDocument%%Created%=10.03.1997
```

## Logical Expressions and Expressions

Query files which are passed to the commands **StartArchiveRequest**, **StartRegRequest**, and **StartDocRequest** result in a hit list of enaio® objects. These query files can contain logical expressions and expressions used to query against the indexing of objects.

### Clauses

Logical expressions follow this syntax:

```
KLAUSEL1=object@field=value
```

A query lets you pass more than one logical expression. To do so, logical expressions need to be numbered sequentially, e.g.:

```
KLAUSEL1=...
KLAUSEL2=...
...
KLAUSEL#N#=#
```

The logical linking of logical expressions is the operator **AND**, i.e. additional logical expressions allowed, limiting the hit list.

**Example** for a logical expression:

```
KLAUSEL1=InboundDocument@Created=10.03.1997
```

In this case **InboundDocument** is the object name and **template** is the name of a field of the object **InboundDocument**. All inbound documents with the value **10.03.1997** in the field **Created** are returned.

Note: Internal object and field names can also be used.

### Expressions

Expressions have a similar syntax. The following forms are possible:

- |         |   |
|---------|---|
| 1. Form | AUSDRUCK1=Object@DBField^OP^Value                         |
| 2. Form | AUSDRUCK1=Object@FieldNo^OP^Value                         |
| 3. Form | AUSDRUCK1=Object@DBField^OP^Value~BoolOP~FieldNo^OP^Value |

**Object** stands for the object name, **DBField** for the column name of the requested field in the database, **OP** stands for the relational operator, **Value** for the field value. **BoolOP** is a logical link of the different relational expressions. **^** separates relational values from operators and **~** separates Boolean expressions from Boolean operators.

A query lets you pass more than one logical expression. To do so, logical expressions need to be numbered sequentially, e.g.:

```
AUSDRUCK1=...
AUSDRUCK2=...
...
AUSDRUCKN=#
```

The logical linking of expressions is the operator **AND**, i.e. additional expressions allowed, limiting the hit list.

**Example 1** for an expression:

```
AUSDRUCK1=InboundDocument@Date1^6^10.03.2000
```

As in the logical expression, **inbound document** indicates the object name. **Date1** is the database column name of the field **Created**. The column name can be determined using the commands **GetObjectFields** and **GetObjectFieldsEx**. All inbound documents are returned that contain a value of greater or equal value to **10.03.1997** in the database field **Date1**.

**Example 2** for an expression:

```
AUSDRUCK1=Account@1100^3^4711
```

All **Account** objects with an object ID of less than 4711 are returned.

**Example 3** for an expression:

```
AUSDRUCK1=Account@1100^3^4711~0~1100^4^0815
```

All **Account** objects with an object ID between 4711 and 0815 are returned. Complex expressions with Boolean operators must be grouped with (and) to assure the unique meaning of the expression.

**Examples for a location preset**

Example: Document in folder

```
AUSDRUCK1=Account@1130^1^815~0~1133^1^0
```

Using this expression in a document query only returns documents from directly under the folder level. **1130^1^815** defines the folder with the ID 815 as the parent folder. **1133^1^0** specifies that the searched registers have no parent register.

Example: Register in folder

```
AUSDRUCK1=Register@1121^1^815~0~1122^1^0
```

Using this expression in a register query only returns registers from directly under the folder level. **1121^1^815** defines the folder with the ID 815 as the parent folder. **1122^1^0** specifies that the searched registers have no parent register.

Example: 'register in register'

```
AUSDRUCK1=Register@1122^1^9911
```

This expression defines the parent register for a register query. Note: if the parent register is of a different kind, the expression will contain the register type of the requested register as an object. This is a list of **field names/field numbers**, operators (**OP**) and Boolean operators (**BoolOP**)

**Field name/Field no.**

Field name	Field no.	Description
<b>Folder</b>		
STAMM_ID	1000	Folder index
STAMM_TIME	1001	Creation time
STAMM_LINKS	1002	Number of links to this folder
<b>Objects</b>		
OBJECT_ID	1100	Document index
OBJECT_COUNT	1101	Number of document files
OBJECT_FLAGS	1102	Archiving status: 0 = archived 1 = archivable 2 = not archivable 4 = error in pages 8 = no pages 16 = archived but no slides 32 = not archived and not visible for archivist
OBJECT_AVID	1103	Name of archivist
OBJECT_AVDATE	1104	Archiving date
OBJECT_CRID	1105	Name of creator
OBJECT_CRDATE	1106	Attachment date
OBJECT_TIME	1107	Time stamp (date and time) of creation
OBJECT_MAIN	1108	Main document type
OBJECT_CO	1109	Secondary document type
OBJECT_MEDDOCID	1110	Media index of document
OBJECT_MEDDIAID	1111	Media index of slide
OBJECT_MEDDOCNA	1112	Path to document
OBJECT_MEDDIANA	1113	Path to slide
OBJECT_LINKS	1114	Number of links to this document
OBJECT_VERID	1115	Index of original variant. Special values: 0=no variants, 1=this is the original document
OBJECT_LOCKUSER	1116	Index of the user who locked the document. Special values: 0=not locked, 1=swapped
<b>Register</b>		
REG_ID	1120	Index of register
REG_STAID	1121	Index of the folder in which the register is located
REG_PARID	1122	Index of the register in which the register is located
<b>Folder-Document-Relation</b>		
SDSTA_ID	1130	Index of the folder in which the document is located
SDOBJ_ID	1131	Document index
SDOBJTYPE	1132	Object type (main type/sub type) of the document
SDREG_ID	1133	Index of the register in which the document is located
SDDEL	1134	Delete flag
SDDTIME	1135	Insertion time
<b>Portfolio Document Relation</b>		
MDDEL	1140	Delete flag
MDTIME	1141	Creation time
MDMAP_ID	1142	Portfolio ID
MDSTA_ID	1143	Folder ID
MDOBJ_ID	1144	ID of the object
MDOBJTYPE	1145	Object type
MDMOD	1146	Main object type (if without type)
MDIN	1147	Entry time (not used)
MDOUT	1148	Exit time (not used)
MDCOUNT	1149	Number of pages (if without type)
MSEND	1150	Sender (not used)
<b>Portfolio</b>		
MAPDEL	1160	Delete flag
MAPTIME	1161	Creation time
MAP_ID	1162	Portfolio ID
MAPCR_ID	1163	Name of creator
MAPCRDATE	1164	Attachment date
MAPRE_ID	1165	Name of recipient
MAPTHEME	1166	Theme of portfolio
MAPTYPE	1167	Portfolio type (not used)
<b>User</b>		
USER_ID	1170	User ID
USER_SUPER	1171	0=no supervisor; 1=supervisor (since 3.00 SP02 this is a combination of the rights flags -> see new rights system)
USER_USER	1172	User name
USER_PASSWORD	1173	Encoded user password
USER_STATION	1174	Workstation name

USER_LOGIN	1175	Login time stamp
------------	------	------------------

**OP List**

OP	SQL operator
1	= (for exact searches, e.g. ^1^899999) LIKE (when searching for strings with placeholders, e.g. ^1^*.pdf)
2	!=
3	<
4	>
5	<=
6	>=
7	In
8	Ex

**BoolOP List**

BoolOP	SQL operator
0	AND
1	OR
2	NOT

Note: Internal object names can also be used.

**Wildcard in Query Files**

These wildcards can be used in query files:

Placeholders	Reference
#COMPUTER-GUID#	GUID of the logged-on user's computer
#COMPUTER-NAME#	Name of the logged-in user's computer
#COMPUTER-IP#	IP address of the logged-on user's computer
#ANLEGER#	Link to the basic parameter field 'Creator'
#ANLEGEDATUM#	Link to the basic parameter field 'Date of creation'
#ARCHIVAR#	Link to the basic parameter field 'Archivist'
#ARCHIVIERUNGSDATUM#	Link to the basic parameter field 'Archiving date'
#BENUTZER#	Name of the logged-on user
#BESITZER#	Link to the basic parameter field 'Owner' which contains the owner's GUID
#DATUM#	Current date

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## Code Examples

### VB and VBA

There are two equivalent values to address the enaio® COM object under VB and VBA.

If enaio® client is processing time-consuming operations, the client application may not react to parallel queries using the COM interface as promptly as one would expect. The 1044 (WM\_USER + 20) Windows message with WPARAM 30 (SendMessage API function) may be used to check the client application's system load beforehand. The value 1 is returned if system load is high; otherwise the return value is 0. It is recommended to run the system load check before COM functions are called and, in case system load is high, to reduce it.

The following examples were created in VB version 6 and Office 97 VBA.

#### 1. Dimensioning a variable as 'New optimal\_AS.Application'.

The enaio® object can only be addressed in such a way if the reference pointing to it was activated in the VB project. An important advantage of this method is that the enaio® COM object is known to the Intellisense technology of VB and VBA including all parameters, and that the Intellisense lists are displayed. In the following example the license registration for the COLD-module is performed and according to the file **InsInA.txt** a cabinet is created which is instantly deleted. At the end of the **Main** routine an error handling routine can be found which returns the error text along with the COM error number of enaio®.

```
Dim MyAX As New optimal_AS.Application
Dim HelpInt as Integer
Dim InsertFile as String, DeleteString as String, ErrorString as String
Dim AxID as Long, AxObjectType as Long
Sub Main()
    HelpInt = COMLicLogin("COL")
    If HelpInt <> 0 Then Goto ErrorHandler
    InsertFile = App.Path & "\InsInA.txt"
    HelpInt = COMInsertArc(InsertFile)
    If HelpInt <> 0 Then Goto ErrorHandler
    DeleteString = CStr(AxID) & "," & CStr(AxObjectType)
    HelpInt = COMDelArc(DeleteString)
    If HelpInt <> 0 Then Goto ErrorHandler
    Exit Sub
ErrorHandler:
    ErrorString = MyAX.GetLastError
    msgbox ErrorString
End Sub

Function COMLicLogin(LicStr as String) as Integer
    COMLicLogin = MyAX.LicLogin(LicStr)
End Function

Function COMInsertArc(File as String) as Integer
    COMInsertArc = MyAX.InsertIntoArchive(File, AxID, AxObjectType)
End Function

Function COMDelArc(DelStr as String) as Integer
    COMDelArc = MyAX.DeleteFromArchive(DelStr)
End Function
```

#### 2. Creating an Object with the 'CreateObject' Command.

This method can also be used with VB script; not the first method as project references are required which are not possible. This disadvantage of this method is that VB and VBA do not provide Intellisense support.

```
Dim MyAX As Object
....
rest like in the previous example
....
Sub Main()
    Set MyAX = CreateObject("Optimal_AS.Application")
    ....
    rest like in the previous example
    ....
end Sub
Function COMLicLogin(LicStr as String)
    COMLicLogin = MyAX.LicLogin(LicStr)
End Function
....
rest like in the previous example
....
```

**Note:** As there are already many VB projects with COM integration, the enaio® COM object is still called **optimal\_AS** for compatibility reasons. The **CreateObject** method is suggested in order to create an optimal\_AS object. Internally VB saves the GUID of the registered object. If you work with project references and import a new version of enaio®, it may happen that the new version creates a new object GUID. As a result, the object cannot be created and the program quits with an error message.

In order to use **Intellisense** technology in VB projects, you can use new development methods, i.e. project reference with **New** dimensioning of the object as well as creating objects with **CreateObject** in the source code. Having finished the development, comment out the **New** dimensioning.

## Object Main Types

For some COM commands a document main type must be specified. The following settings are possible.

Main type no.	Main type name
1	X-Document (grayscale image)
2	D-Document (B/W image)
3	P-Document (color image)
4	W-Document (Windows document)
5	M-Document (video document)
6	E-Document (E-mail)
7	XML-Document (XML)
8	Container document

# All COM Commands

## ActivateApp

<b>Definition</b>	ActivateApp (short nShow)						
<b>Type</b>	Method						
<b>Description</b>	Displays the application window or hides it						
<b>Parameters</b>	<table><tr><td><b>nShow</b></td><td><b>1</b></td><td>Show</td></tr><tr><td></td><td><b>0</b></td><td>Hide</td></tr></table>	<b>nShow</b>	<b>1</b>	Show		<b>0</b>	Hide
<b>nShow</b>	<b>1</b>	Show					
	<b>0</b>	Hide					
<b>Return value</b>							
<b>Comment</b>							
<b>Options</b>							
<b>Example</b>							

## AdjustRetention

<b>Definition</b>	AdjustRetention	(long long Variant	lObjectID, lObjectType, varRetentionDate)
<b>Type</b>	Method		
<b>Description</b>	Adjusts the retention date to the scheduled retention date.		
<b>Parameters</b>	<b>lObjectID</b> <b>lObjectType</b> <b>varRetentionDate</b>	object ID object type the adjusted retention date is returned in the format	<b>DD.MM.YYYY</b>
<b>Return value</b>	<b>0</b> <b>-1</b> <b>-7</b> <b>-22</b> <b>-38</b> <b>-40</b> <b>-112</b>	retention date successfully adjusted retention date could not be adjusted object type unknown an object with the specified ID does not exist invalid document type (a folder or register type was specified) invalid parameter passed the specified object has not been archived (yet).	

**Comment** This function can only be used to change already archived documents. Error texts can be determined using GetLastError().

### Options

### Example

```
Dim a As Object
Dim retDate

Set a = CreateObject("optimal_as.application")

a.AdjustRetention lObjectID, lObjectType, retDate
```

## AppBrowserSession

### Definition

```
[AppBrowserSession]
URL=file:///C:/test1.html
MODE=0
TIMER=30
```

### Type

### Description

Provides a Chromium browser session which can be used as a connection between the client and a Web server. In the client you can then trigger actions using the JavaScript interface of the browser. If a URL is configured, it is already loaded when starting the client. Use the as.cfg to configure the browser session.

### Parameters

**URL** URL to load. If not specified, no action is triggered. Default: not specified

**MODE** Specify the situation when to load the given URL.  
MODE=1 -> On context change  
MODE=0 -> No action

**TIMER** Specify (in seconds) when to load the given URL. Valid values: 0 to 3600  
Standard: TIMER=0 -> no timer

### Return value

### Comment

### Options

### Example

## ApplicationLogin

<b>Definition</b>	ApplicationLogin (String strUser)
<b>Type</b>	Method
<b>Description</b>	Allows a new user to log in.
<b>Parameters</b>	<b>strUser</b> string with user name and password, separated by a '@'
<b>Return value</b>	1 no error
<b>Comment</b>	If the <b>DIALOG</b> string is passed, in any case the login dialog will open. This is only available if password verification is not handled with <b>Novell NetWare</b> but with a user dialog of the archiving system.
<b>Options</b>	
<b>Example</b>	

## CalcFileDigest

<b>Definition</b>	CalcFileDigest (string FileName, VARIANT* vRetDigest)
<b>Type</b>	Method
<b>Description</b>	Determines the hash value (digest) of the specified file
<b>Parameters</b>	<b>FileName</b> the file name <b>vRetDigest</b> the determined hash value (digest)
<b>Return value</b>	0 no error -1 hash value could not be determined -32 the specified file does not exist
<b>Comment</b>	
<b>Options</b>	
<b>Example</b>	

## CheckInDocument

<b>Definition</b>	CheckInDocument	(long long String	IDocID, IDocType, strSourcePath)
<b>Type</b>	Method		
<b>Description</b>	Checks in again a document that was checked out for editing.		
<b>Parameters</b>	<b>IDocID</b> <b>IDocType</b> <b>strSourcePath</b>	Document ID Document type Path (without file name) to the file to be checked in	
<b>Return value</b>	0 -7 -53 -54 -55 -56 -57	no error unknown document type document was not checked out document was not checked out for external editing file could not be copied to cache directory document name could not be determined in cache source file does not exist	Error text can be determined using <b>GetLastError()</b> .

### Comment

### Options

### Example

## CheckLicence

<b>Definition</b>	CheckLicence (String strModuleName)
<b>Type</b>	Method
<b>Description</b>	Verifies if a specific module is licensed for the current workstation
<b>Parameters</b>	<b>strModuleName</b> module name
<b>Return value</b>	<b>0</b> is licensed <b>-1</b> is not licensed
<b>Comment</b>	
<b>Options</b>	
<b>Example</b>	

# CheckObjectAccess

<b>Definition</b>	CheckObjectAccess	(long long long long short	(lObjectID, lObjectType, lDesiredAccess, lFlags, nShowError)
<b>Type</b>	Method		
<b>Description</b>	Verifies access rights for the specified object		
<b>Parameters</b>	<b>lObjectID</b>	object ID	
	<b>lObjectType</b>	object type	
	<b>lDesiredAccess</b>	<b>0</b>	Read index data
		<b>1</b>	Write index data
		<b>2</b>	Delete object
		<b>3</b>	Output object (e.g. print)
		<b>4</b>	Write object
		<b>5</b>	Check out status
		<b>6</b>	Archive status
		<b>7</b>	Specifies whether the object is in the workflow tray (return value= <b>1</b> ) or not ( <b>0</b> )
		<b>8</b>	Specifies whether the object is marked for deletion. Return value: <b>0</b> , object not marked for deletion <b>1</b> , object marked for deletion <b>-22</b> unknown object
	<b>lFlags</b>	since 4.20 SpII the value is set to <b>1</b> , in order to request the correct user rights, even when the security system is disregarded for this object.	
	<b>nShowError</b>	if not equal to <b>0</b> an error dialog is displayed	
<b>Return value</b>	<b>0</b>	user has no access right	
	<b>1</b>	user has access right	
	<b>-1</b>	unknown right	
	<b>-7</b>	document type unknown	
	<b>-22</b>	unknown object	
	<b>-46</b>	unknown user	
	<b>-51</b>	document contains no files	
	<b>-52</b>	object already archived	
	<b>-58</b>	document already loaned	
	<b>-59</b>	document checked out by other user editing	
	<b>-60</b>	object is not a W-Document	

**Comment**

**Options**

**Example**

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## CheckOutDocument

<b>Definition</b>	CheckOutDocument	(long long String Variant*	IDocID, IDocType, strPath, varRetDocName)
<b>Type</b>	Method		
<b>Description</b>	Checks out a document for external editing.		
<b>Parameters</b>	<b>IDocID</b>	Document ID	
	<b>IDocType</b>	Document type	
	<b>strPath</b>	Path to where the document will be checked out	
	<b>varRetDocName</b>	here, the full path and file name for the checked out document is displayed	
<b>Return value</b>	<b>0</b>	no error	
	<b>-1</b>	the document could not be checked out	
	<b>-7</b>	unknown document type	
	<b>-51</b>	document contains no files	
	<b>-52</b>	object already archived	
	<b>-58</b>	the document was already checked out externally	
	<b>-59</b>	the document was checked out by another user	
		Error text can be determined using <b>GetLastError()</b> .	

### Comment

### Options

### Example

**Notes:** The checked-out file must not be renamed.

## ClearSignatureProperties

<b>Definition</b>	ClearSignatureProperties()
<b>Type</b>	Method
<b>Description</b>	Deletes all properties that were previously defined with <b>SetSignatureProperty</b> .
<b>Parameters</b>	
<b>Return value</b>	
<b>Comment</b>	It is recommended that this function is executed right after a digital signature.
<b>Options</b>	
<b>Example</b>	

## CloseAllWindows

<b>Definition</b>	CloseAllWindows ()
<b>Type</b>	Method
<b>Description</b>	Closes all child windows of the enaio® client.
<b>Parameters</b>	
<b>Return value</b>	
<b>Comment</b>	
<b>Options</b>	
<b>Example</b>	

## CloseOperationID

<b>Definition</b>	CloseOperationID (long lObjectID, long lObjectType)
<b>Type</b>	Method
<b>Description</b>	Closes a window opened by <b>OpenObjectID</b> .
<b>Parameters</b>	<b>lObjectID</b> object ID <b>lObjectType</b> object type
<b>Return value</b>	0 no error -11 object ID invalid Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	
<b>Options</b>	
<b>Example</b>	

## ConvertImage

<b>Definition</b>	ConvertImage	(String String short short short short	strSource, strDestination, nFormat, nCompression, nBitsPerPixel, nFlags)
<b>Type</b>	Method		
<b>Description</b>	This function converts an image file into the specified format.		
<b>Parameters</b>	<b>strSource</b>	source file	
	<b>strDestination</b>	destination file	
	<b>nFormat</b>	destination format, see GetImageType	
	<b>nCompression</b>	compression method or loss factor for JPEG-format, 1 recommended	
	<b>nBitsPerPixel</b>	color depth 1, 2, 4, 8, 16, 24 or 0 for color depth of the source file	
	<b>nFlags</b>	not used	
<b>Return value</b>	0	file was successfully converted	
	<> 0	error	
<b>Comment</b>			
<b>Options</b>			
<b>Example</b>			

# CopyObject

<b>Definition</b>	CopyObject	(long long long long short Variant	(lObjectID, lObjectType, lFolderID, lRegisterID, nFlags, varRetObjectID)
<b>Type</b>	Method		
<b>Description</b>	Creates a copy of an object including multiple fields and document.		
<b>Parameters</b>	<b>lObjectID</b>	object ID of the object to be copied	
	<b>lObjectType</b>	object type of the object to be copied	
	<b>lFolderID</b>	ID of the destination folder; set the value to 0 to copy a folder.	
	<b>lRegisterID</b>	ID of the destination register, 0, unless to be copied into a register	
	<b>nFlags</b>	0 copy indexing and document 1 copy only indexing	
	<b>varRetObjectID</b>	here the ID of the new object is returned	
<b>Return value</b>	0 no error -11 document ID unknown -22 object type unknown -23 destination register unknown -47 user has no write access or is not allowed to create new objects. -61 no definite assignment of the object possible. Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	Creates a copy of an object including multiple fields and document. If the object is part of the version administration, the current version will be copied. This function can also be used to copy folders and registers. In this case the folder or register contents are not copied.		
<b>Options</b>			
<b>Example</b>			

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## CreateDocumentLink

<b>Definition</b>	CreateDocumentLink (long long long long IObjectID, IObjectType, ITargetID, ITargetType)																				
<b>Type</b>	Method																				
<b>Description</b>	Creates a new reference to a document																				
<b>Parameters</b>	<table> <tr> <td><b>IObjectID</b></td> <td>object index</td> </tr> <tr> <td><b>IObjectType</b></td> <td>object type</td> </tr> <tr> <td><b>ITargetID</b></td> <td>index of the destination folder/register</td> </tr> <tr> <td><b>ITargetType</b></td> <td>object type of target folder/register</td> </tr> </table>	<b>IObjectID</b>	object index	<b>IObjectType</b>	object type	<b>ITargetID</b>	index of the destination folder/register	<b>ITargetType</b>	object type of target folder/register												
<b>IObjectID</b>	object index																				
<b>IObjectType</b>	object type																				
<b>ITargetID</b>	index of the destination folder/register																				
<b>ITargetType</b>	object type of target folder/register																				
<b>Return value</b>	<table> <tr> <td>-1</td> <td>reference could not be created</td> </tr> <tr> <td>-3</td> <td>cabinet unknown</td> </tr> <tr> <td>-5</td> <td>register unknown</td> </tr> <tr> <td>-7</td> <td>document type unknown</td> </tr> <tr> <td>-13</td> <td>register identification unknown (if the destination type is a register)</td> </tr> <tr> <td>-14</td> <td>folder identification unknown (if destination type is a folder)</td> </tr> <tr> <td>-20</td> <td>specified object type is not a document type</td> </tr> <tr> <td>-22</td> <td>unknown object type</td> </tr> <tr> <td>-33</td> <td>document type invalid (does not match cabinet)</td> </tr> <tr> <td>-89</td> <td>object must not be created in the destination folder/register</td> </tr> </table> <p>Error text can be determined using <b>GetLastError()</b>.</p>	-1	reference could not be created	-3	cabinet unknown	-5	register unknown	-7	document type unknown	-13	register identification unknown (if the destination type is a register)	-14	folder identification unknown (if destination type is a folder)	-20	specified object type is not a document type	-22	unknown object type	-33	document type invalid (does not match cabinet)	-89	object must not be created in the destination folder/register
-1	reference could not be created																				
-3	cabinet unknown																				
-5	register unknown																				
-7	document type unknown																				
-13	register identification unknown (if the destination type is a register)																				
-14	folder identification unknown (if destination type is a folder)																				
-20	specified object type is not a document type																				
-22	unknown object type																				
-33	document type invalid (does not match cabinet)																				
-89	object must not be created in the destination folder/register																				

### Comment

### Options

### Example

# CreateMimeFile

**Definition** CreateMimeFile (String strFrom, String strTo, String strCC, String strBCC, String strSubject, String strBody, String strAttachments, String strCreationTime, Variant\* varRetFilename)

**Type** Method

**Description** Creates a Mime-file

**Parameters**

<b>strFrom</b>	sender address
<b>strTo</b>	recipient
<b>strCC</b>	CC
<b>strBCC</b>	BCC
<b>strSubject</b>	subject
<b>strBody</b>	text
<b>strAttachments</b>	attached files separated by a pipe character
<b>strCreationTime</b>	creation time in the format: day.month.year hour:minute
<b>varRetFilename</b>	full path and file name of created Mime file

**Return value**

0	no error
-1	Mime-file could not be created

**Comment** If required, the created file can be deleted by the requesting application.

**Options**

**Example**

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## DecodeIMAPFile

<b>Definition</b>	DecodeIMAPFile	(string Variant Variant Variant Variant Variant Variant Variant Variant)	strFilename, strMailDate, strFrom, strTo, strCC, strBCC, strSubject, strBody, strAttachments)
<b>Type</b>	Method		
<b>Description</b>	Decodes an IMAP file and returns its content.		
<b>Parameters</b>	<b>strFilename</b>	full path and file name of the IMAP file File	
	<b>strMailDate</b>	The sending date of the e-mail is returned returned.	
	<b>strFrom</b>	here the name of the sender is returned.	
	<b>strTo</b>	the recipient list (separated by semicolon) is returned here.	
	<b>strCC</b>	the CC list (separated by semicolon) is returned here.	
	<b>strBCC</b>	the BCC list (separated by semicolon) is returned here.	
	<b>strSubject</b>	the subject is returned here.	
	<b>strBody</b>	the e-mail content is returned here.	
	<b>strAttachments</b>	the attachments are returned here, separated with semicolons	
<b>Return value</b>	0	if no error	
	-1	if the file could not be decoded	
<b>Comment</b>			

## CreateNewDocShare

<b>Definition</b>	CreateNewDocShare (long long string string string IIdent, IType, Rights Users Info)
<b>Type</b>	Method
<b>Description</b>	Creates new shares of the specified document for the specified users
<b>Parameters</b>	<b>IIdent</b> Document ID <b>IType</b> Document Type <b>Rights</b> Rights set as defaults (RWXU) <b>Users</b> User ID's of the user who specified Document to be released, separated by semicolon <b>Info</b> Info text for the release

<b>Return value</b>	<b>0</b> "New release(s) created." <b>-125</b> "A modal dialog is opened." <b>-130</b> "Specified DMS object type is unknown." <b>-131</b> "Specified DMS object type if not a document." <b>-132</b> "Only 'RXWU' rights are permitted." <b>-133</b> "The specified document does not exist or was deleted." <b>-134</b> "Document shares not possible." <b>-135</b> "The system role required to share documents is not present." <b>-136</b> "At least one specified user does not exist." <b>-137</b> "Cancellation by user." <b>-138</b> "Server reports error: "
---------------------	---

Error exact text can be determined using **GetLastError()**.

**Comment** The 'Rights', 'Users' and 'Info' parameters can be empty.

### Example

```

dim Application : set Application =
createobject("optimal_AS.application")
Dim sRet

Application.ActivateApp 1

sRet = Application.CreateNewDocShare(590, 262146, "WX", "416;15475",
"Aufruf aus COM-Methode")

if (sRet <> 0) then
    MsgBox Application.GetLastError() & " (" & sRet & ")"
else
    MsgBox "Neue Freigabe(n) angelegt!"
end if

set Application = nothing

```

## DeleteFromArchive

<b>Definition</b>	DeleteFromArchive (String strParam)
<b>Type</b>	Method
<b>Description</b>	Deletes the specified folder or the specified document.
<b>Parameters</b>	<b>strParam</b> string with object index and object type or file name of a file with the following structure:*

```
Objectindex1,Objecttype1\r\n
Objectindex2,Objecttype2\r\n
...
Objectindexn,Objecttypen\r\n
```

\* \r\n stands for a line break, CR and LF.

<b>Return value</b>	<p><b>0</b> no error</p> <p><b>-1</b> handoff string incorrect</p> <p><b>-11</b> invalid document identifier</p> <p><b>-41</b> handoff string empty</p> <p><b>-44</b> The logged-in user is not authorized to delete at least one of the document types in the file.</p> <p><b>-45</b> The logged-in user has no access to one or more files in the requested document type.</p> <p><b>-69</b> The specified handoff file is empty.</p> <p><b>-77</b> The server cannot delete the specified object. (further information visible in the server logging)</p> <p><b>-78</b> The document is being used by one or more workflow processes and cannot be deleted.</p> <p>Error text can be determined using <b>GetLastError()</b>.</p>
---------------------	---

<b>Comment</b>	<p>This function deletes the specified object <b>without prompting</b> if the user who is logged on to the archive system has sufficient rights to delete the object.</p> <p>It is also possible when handing over a string with object ID and type to delete several objects with one request. The string needs to have the following structure.</p>
----------------	---

```
object ID1,object type1 object ID2,object type2 object IDn
object typen
```

### Options

### Example

## DoPrefetch

<b>Definition</b>	DoPrefetch	(Variant Variant	IObjectID, IObjectType)
<b>Type</b>	Method		
<b>Description</b>	Performs a prefetch for the given object.		
<b>Parameters</b>	<b>IObjectID</b>	object ID as variant	
	<b>IObjectType</b>	object type as variant	
<b>Return value</b>	0 no error <> 0 error Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>			
<b>Options</b>			
<b>Example</b>			

## ExecuteRequest

<b>Definition</b>	ExecuteRequest (String strRequestName)
<b>Type</b>	Method
<b>Description</b>	Performs a saved query.
<b>Parameters</b>	<b>strRequestName</b> name of the saved query
<b>Return value</b>	<b>0</b> no error <b>-1</b> no saved queries found <b>-2</b> a query with the specified name does not exist Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	Displayed by the enaio® client.

If the saved query has variables, they can be specified in the following form:

```
VAR1=Content1;VAR2=Content2;...VARn=Contentn
```

Values of static variables can be placed in a request too:

```
STAT1=Content1;STAT2=Content2;...STATn=Contentn
```

A saved query with variables opens a request page if not all variables have been specified in the request; specified fields are already filled in.

In any case the saved query can be 'forced' to display the request form by using the 'OPT1=1' switch.

<b>Options</b>	<b>OPT1=1</b> Forces the query form to be opened
----------------	---

<b>Examples</b>	Request for the saved query 'Test' which contains static variables:
-----------------	---

```
ExecuteRequest "test STAT1=Hallo;VAR1=Bert*"
```

Request for the saved query 'Test' which contains static variables and that opens the request page in any case:

```
ExecuteRequest "test OPT1=1;STAT1=Hello;VAR1=Bert*"
```

All fields assigned with variables are then filled with their corresponding contents.

Request for the saved query 'Test' with variables handoff

```
Test VAR1=199*;VAR2=Bert*
```

| Note: The request data is not verified. SQL errors may occur if e.g. a date is expected and text is specified. |

## ExecuteRequestEx

<b>Definition</b>	ExecuteRequestEx (String strRequestName, String strParams)
<b>Type</b>	Method
<b>Description</b>	As with <b>ExecuteRequest</b> , but the parameters for the saved query need to be specified additionally.
<b>Parameters</b>	<b>strRequestName</b> name of the saved query <b>StrParams</b> parameter for this query (see <b>ExecuteRequest</b> )
<b>Return value</b>	0 no error -1 no saved queries found -2 a query with the specified name does not exist Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	In contrast to <b>ExecuteRequest</b> this function can also process saved queries with names containing spaces.  See also <b>ExecuteRequest()</b>
<b>Options</b>	
<b>Example</b>	

## FindObjectType

<b>Definition</b>	FindObjectType(long lObjectID)
<b>Type</b>	Method
<b>Description</b>	Determines the object type corresponding with a specified object ID
<b>Parameters</b>	<b>lObjectID</b> object ID for which the object type is to be determined
<b>Return value</b>	<b>String</b> with object type <b>Empty string</b> if no object type could be determined.
<b>Comment</b>	In rare cases it may occur that the object type cannot be determined unambiguously. In such case the result string will contain all object types consecutively separated by a comma.
<b>Options</b>	
<b>Example</b>	

## FindObjectTypeEx

<b>Definition</b>	FindObjectTypeEx	(long long VARIANT* IObjectID IType, vRetObjTypes)
<b>Type</b>	Method	
<b>Description</b>	Determines the object type corresponding with a specified object index	
<b>Parameters</b>	<b>IObjectID</b>	object ID for which the object type is to be determined
	<b>IType</b>	type of the object type to be determined: <b>0</b> document <b>1</b> folder <b>2</b> register
	<b>vRetObjTypes</b>	here the determined object type is returned
<b>Return value</b>	<b>0</b>	no error
	<b>-26</b>	object index invalid
	<b>-40</b>	specified type invalid
<b>Comment</b>	By specifying the type this function allows quicker finding the object type. In rare cases it may occur that the object type cannot be determined unambiguously. In such case the result string will contain all object types consecutively separated by a comma.	

### Options

### Example

## FreeDocument

<b>Definition</b>	FreeDocument (long lObjectID)
<b>Type</b>	Method
<b>Description</b>	Unblocks the specified W-Document.
<b>Parameters</b>	<b>lObjectID</b> index of the object to be unblocked
<b>Return value</b>	>0    number of documents that could not be checked in 0    no error -1    document could not be checked in by the archive server -2    document locked
<b>Comment</b>	If a <b>0</b> is specified for the object ID, all locked documents are unlocked.
<b>Options</b>	
<b>Example</b>	

## FreeDocumentEx

<b>Definition</b>	FreeDocumentEx (String strDocuments)
<b>Type</b>	Method
<b>Description</b>	Loans the specified documents with an additional thread.
<b>Parameters</b>	<b>strDocuments</b> string with IDs, separated by semicolons, of the objects to be loaned out.
<b>Return value</b>	0
<b>Comment</b>	This function opens a thread to loan out objects i.e. it does not wait until all objects were loaned out. If a 0 is specified for the object ID, all locked documents are unlocked.

### Options

### Example

## GenerateColdFiles

<b>Definition</b>	GenerateColdFiles	(String String String VARIANT* strFile1, strFile2, strToken, vRetFileList)
<b>Type</b>	Method	
<b>Description</b>	Creates image objects from ASCII-COLD import files.	
<b>Parameters</b>	<b>strFile1</b>	either a data file or a control file
	<b>strFile2</b>	either a data file or a control file
	<b>strToken</b>	separator for file list
	<b>vRetFileList</b>	list of file names of the created images, separated by the characters specified by strToken.
<b>Return value</b>	<b>0</b>	no error
	<b>-83</b>	sub directory could not be created for the Cold-files
	<b>-84</b>	image could not be created
	<b>-85</b>	file could not be moved to the destination directory
<b>Comment</b>		
<b>Options</b>		
<b>Example</b>		

## GenerateOSFile

<b>Definition</b>	GenerateOSFile	(long long	LObjectID, LObjectType)
<b>Type</b>	Method		
<b>Description</b>	Creates an OS file from the specified parameters.		
<b>Parameters</b>	<b>LObjectID</b>	object ID	
	<b>LObjectType</b>	object type	
<b>Return value</b>	<b>Empty string</b> if error <b>file name</b> of the created OS-file Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	If required, the created OS-file will be deleted by the requesting program.		
<b>Options</b>			
<b>Example</b>			

## GetActiveDocument

<b>Definition</b>	GetActiveDocument (String strDocName, BOOL bOpen, VARIANT vRetVal)
<b>Type</b>	Method
<b>Description</b>	Delivers the COM object with the optimal_AS: Active Document.
<b>Parameters</b>	<p>strDocName name of the Active Document, as specified in <i>as.cfg</i>.</p> <p>bOpen <b>TRUE</b> if the document has to be opened and is not open already.</p> <p>vRetVal here the error number is returned</p>
<b>Return value</b>	<p>0 no error</p> <p>-70 no Active Document with this name could be found</p> <p>-72 no optimal_AS: Active Document</p> <p>Error text can be determined using <b>GetLastError()</b>.</p>
<b>Comment</b>	
<b>Options</b>	
<b>Example</b>	

## GetAllArchives

**Definition** GetAllArchives()

**Type** Method

**Description** Determine all cabinets

**Parameters**

**Return value** String with all available cabinets

**Comment** The return string has the following format:

```
CABINET1, OBJECTTYPE1; CABINET2, OBJECTTYPE2; ... ;  
CABINETn, OBJECTTYPEn
```

**Options**

**Example** Source code example:

```
Helpstr=MyAX.GetAllArchives()
```

**Helpstr** can contain the following string e.g.:

```
Customer,1;Patient,2
```

## GetAllOpenFolders

**Definition** GetAllOpenFolders()

**Type** Method

**Description** provides a list with IDs and types of all open folders. If registers are opened in folders, their IDs and types will also be specified.

**Parameters**

**Return value** string with a list of the IDs and types in the following form:

```
Folder or register ID, folder or register type;folder or  
register ID, folder or register type;.....
```

**Comment**

**Options**

**Example**

## GetCurrentResultList

<b>Definition</b>	GetCurrentResultList (VARIANT* pstrItems)
<b>Type</b>	Method
<b>Description</b>	Returns the items of the current hit list.
<b>Parameters</b>	<b>pstrItems</b> here the indexes and object types of the objects displayed in the current hit list are being returned.

```
ObjectID1, ObjecttypeI1; ObjectID2, ObjecttypeI2; ...;  
ObjectIDn, ObjecttypeIn
```

<b>Return value</b>	If higher than <b>0</b> , the return value corresponds to the number of returned objects or if lower than <b>0</b> , it corresponds to an error. -40 incorrect input parameter Error text can be determined using <b>GetLastError()</b> .
---------------------	---

**Comment**

**Options**

**Example**

## GetCurrentSelection

<b>Definition</b>	GetCurrentSelection(VARIANT* varResult)	
<b>Type</b>	Method	
<b>Description</b>	provides the IDs and object types selected in the current hit list	
<b>Parameters</b>	<b>varResult</b> here the indexes and object types are returned in the following form: <table border="1"><tr><td>ObjectID1, Objecttype1; ObjectID2, Objecttype2; . . . . .</td></tr></table>	ObjectID1, Objecttype1; ObjectID2, Objecttype2; . . . . .
ObjectID1, Objecttype1; ObjectID2, Objecttype2; . . . . .		
<b>Return value</b>	The return value corresponds with the number of selected objects.	
<b>Comment</b>		
<b>Options</b>		
<b>Example</b>		

# GetDataID

<b>Definition</b>	GetDataID	(long long short short	(lObjectID, lObjectType, nMode, bWriteToFile)
<b>Type</b>	Method		
<b>Description</b>	Returns object data i.e. field names and values		
<b>Parameters</b>	<b>LObjectID</b>	Index of the object to be opened	
	<b>lObjectType</b>	object type	
	<b>nMode</b>	controls the return value.	
	<b>0</b>	Returns field names and field values	
	<b>1</b>	Returns basic parameters if it is a folder or register	
	<b>2</b>	Returns names and values of multi-fields	
	<b>10</b>	Corresponds to the value <b>0</b> , but returns the internal field names	
	<b>12</b>	Corresponds to the value <b>2</b> , but returns the internal field names	
	<b>bWriteToFile</b>	If the value is <b>0</b> , the object names are returned in a string. If it is 1 they are written into a file and the file name is returned.	

**Return value**      **Result string** or **File name**.  
**Empty string**      if error  
 Error text can be determined using **GetLastError()**.

**Comment**

**Options**

**Example**

For nMode = 0:

```
Fieldlabel = Fieldcontent\r\n
Fieldlabel = Fieldcontent\r\n
```

For nMode = 1:

```
ZEITSTEMPEL = FieldContent\r\n
ANLEGER = Fieldcontent\r\n
ARCHIVAR = FieldContent\r\n
ANGELEGT = Fieldcontent\r\n
ARCHIVIERT = Fieldcontent\r\n
_LOCATIONS = number in SDREL_
```

For nMode = 2 multiple fields of an object:\*

```
Mehrfachfeldname=Seitennummer,Werte\r\n
Mehrfachfeldname=Seitennummer,Werte\r\n
```

\* \r\n stands for a line break, **CR** and **LF**.

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## GetDescription

<b>Definition</b>	GetDescription	(String long String	strShortName, IObjectType, strFieldName)
<b>Type</b>	Method		
<b>Description</b>	Provides the description text from a list field or structure tree for a specified short form.		
<b>Parameters</b>	<b>strShortName</b>	short form that will be provided with the description text	
	<b>IObjectType</b>	object type	
	<b>strFieldName</b>	name of the list or structure tree field	
<b>Return value</b>	<b>String</b>	with the description, <b>empty String</b> if no description is available or errors occurred. Error text can be determined using <b>GetLastError()</b> .	
<b>Comment</b>	This function is only valid for list and structure tree fields.		

### Options

### Example

The field 'month' has a list with the following entries:

```
01|January
02|February
03|March
```

#### Source code example

```
HelpStr=MyAX.GetDescription("02", 65535, "Month")
```

**HelpStr** now has the value "February"

## GetDocTypesFromArchive

**Definition** GetDocTypesFromArchive (String strSchrankName)

**Type** Method

**Description** Determines all document types belonging to a cabinet.

**Parameters** **strSchrankName** string with the cabinet label

**Return value** String with the cabinet's document types.

**Comment** The return string has the following format:

```
DOCUMENT1, OBJECTTYPE1; DOCUMENT2, OBJECTTYPE2 ...  
DOCUMENTn, OBJECTYPEn
```

**Options**

**Example** Source code example:

```
Helpstr=MyAX.GetDocTypesFromArchive
```

Helpstr may contain the following string:

```
IncomingDocument, 131085; OutgoingDocument, 262159; Attribute, 13  
1086
```

## GetEnvironment

<b>Definition</b>	GetEnvironment (short nEnvType)
<b>Type</b>	Method
<b>Description</b>	Determines settings from the archive system
<b>Parameters</b>	<b>nEnvType</b> number of the setting to be determined
<b>Return value</b>	String with the required setting
<b>Comment</b>	<p>Possible values for nEnvType:</p> <ul style="list-style-type: none"> <li><b>0</b>    determines osGetTmpDir</li> <li><b>1</b>    determines osGetAppCWD</li> <li><b>2</b>    determines osGetCfgFileName</li> <li><b>3</b>    determines osGetUserName</li> <li><b>4</b>    determines osGetHomeCWD</li> <li><b>5</b>    determines osGetIniFileName</li> <li><b>6</b>    determines the application name, in case of an OEM version, the OEM name of the client</li> <li><b>7</b>    determines the ID of the current user</li> <li><b>8</b>    determines osGetStationNumber</li> <li><b>9</b>    determines osGetLocalWorkDir</li> <li><b>10</b>   determines FileVersion of the client</li> <li><b>11</b>   determines all group names of the current user. If the user is part of more than one group, the group names are separated by semicolons.</li> <li><b>12</b>   not used</li> <li><b>13</b>   determines a list with the document indexes newly created in the current session. If more than one documents were created, they are listed consecutively separated by commas. E.g. „1234,1235,1236“. If no documents had been created the return value contains "EMPTY"</li> <li><b>14</b>   determines the full user name</li> <li><b>15</b>   determines maximally allowed size of text notes.</li> <li><b>16</b>   determines the e-mail address of the logged in user in case it is known to the system.</li> <li><b>17</b>   determines the remark field for the logged in user.</li> <li><b>18</b>   determines the GUID of the logged in user.</li> <li><b>19</b>   determines GUID of the department of the logged in user.</li> <li><b>20</b>   determines number of failed log ins of a user, after the last successful log in.</li> <li><b>21</b>   determines the time from when the account of the current user is available.</li> <li><b>22</b>   determines the time to when the account of the current user is available.</li> <li><b>23</b>   determines osGetDocLockStation.</li> <li><b>24</b>   determines the GUI language set in the client: "en" for English, "fr" for French, and 'de' for German (default).</li> <li><b>25</b>   determines the ID of the object definition language set in the client: e.g. <b>7</b>-German, <b>9</b>-English.</li> </ul>

### Options

### Example

## GetFilesFromID

<b>Definition</b>	GetFilesFromID (long long String IObjectID, IObjectType, strToken)
<b>Type</b>	Method
<b>Description</b>	Determines the image files belonging to the object. If the images are archived, they are first loaded into the cache.
<b>Parameters</b>	<b>IObjectID</b> index of the object <b>IObjectType</b> object type <b>strToken</b> separator Error text can be determined using <b>GetLastError()</b> .
<b>Return value</b>	<b>String</b> with file name incl. path <b>Empty string</b> if error Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	If multiple pages are part of a document, the file names are listed consecutively separated by a space in the string. If a separator was specified, file names are separated by this character. This makes sense as file names can contain spaces.
<b>Options</b>	
<b>Example</b>	

## GetFilesFromIDEx

<b>Definition</b>	GetFilesFromIDEx	(long long String BOOL VARIANT	(IObjectID, IObjectType, strToken, bWriteProtected, vFileNames)
<b>Type</b>	Method		
<b>Description</b>	Determines the files corresponding with an object.		
<b>Parameters</b>	<b>IObjectID</b>	object ID	
	<b>IObjectType</b>	object type	
	<b>strToken</b>	separator used to separate the names in the return string	
	<b>bWriteProtected</b>	<b>TRUE</b> retrieve document with write protection	
	<b>vFileNames</b>	here the file names are returned	
<b>Return value</b>	<b>0</b>	no error	
	<b>-7</b>	document type unknown	
	<b>-15</b>	document ID unknown	
	<b>-51</b>	document has no files	
	Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	In contrast to the <b>GetFilesFromID()</b> it can be specified if the files are checked out or with write protection.		
<b>Options</b>			
<b>Example</b>			

## GetImageType

<b>Definition</b>	GetImageType (String strSource)
<b>Type</b>	Method
<b>Description</b>	Specifies the image type of the given file.
<b>Parameters</b>	<b>strSource</b> full path and file name of the image file
<b>Return value</b>	<b>&gt; 0</b> file type see comment <b>&lt; 0</b> error <b>-20009</b> file format erroneous

**Comment** The following file types are defined:

PCX(1), GIF(2), TIF(3), TGA(4), CMP(5), BMP(6), FROM\_BUFFER(7), BITMAP(9), JFIF(10), JTIF(11), BIN(12), HANDLE(13), OS2(14), WMF(15), EPS(16), TIFLZW(17), LEAD(20), LEAD1JFIF(21), LEAD1JTIF(22), LEAD2JFIF(23), LEAD2JTIF(24), CCITT(25), LEAD1BIT(26), CCITT\_GROUP3\_1DIM(27), CCITT\_GROUP3\_2DIM(28), CCITT\_GROUP4(29), LEAD\_NOLOSS(30), FILE\_CALS(50), MAC(51), IMG(52), MSP(53), WPG(54), RAS(55), PCT(56), PCD(57), DXF(58), AVI(59), WAV(60), FLI(61), CGM(62), EPSTIFF(63), EPSWMF(64), CMPNOLOSS(65), FAX\_G3\_1D(66), FAX\_G3\_2D(67), FAX\_G4(68), WFX\_G3\_1D(69), WFX\_G4(70), ICA\_G3\_1D(71), ICA\_G3\_2D(72), ICA\_G4(73), OS2\_2(74), PNG(75), PSD(76), RAWICA\_G3\_1D(77), RAWICA\_G3\_2D(78), RAWICA\_G4(79), FPX(80), FPX\_SINGLE\_COLOR(81), FPX\_JPEG(82), FPX\_JPEG\_QFACTOR(83), BMP\_RLE(84), TIF\_CMYK(85), TIFLZW\_CMYK(86), TIF\_PACKBITS(87), TIF\_PACKBITS\_CMYK(88), DICOM\_GRAY(89), DICOM\_COLOR(90), WIN\_ICO(91), WIN\_CUR(92), TIF\_YCC(93), TIFLZW\_YCC(94), TIF\_PACKBITS\_YCC(95), EXIF(96), EXIF\_YCC(97), EXIF\_JPEG(98), AWD(99), FASTEST(100), EXIF\_JPEG\_411(101), PBM\_ASCII(102), PBM\_BINARY(103), PGM\_ASCII(104), PGM\_BINARY(105), PPM\_ASCII(106), PPM\_BINARY(107), CUT(108), XPM(109), XBM(110), IFF\_ILBM(111), IFF\_CAT(112), XWD(113), CLP(114), JBIG(115), EMF(116), ICA\_IBM\_MMR(117), RAWICA\_IBM\_MMR(118), ANI(119), ANI\_RLE(120), LASERDATA(121), INTERGRAPH\_RLE(122), INTERGRAPH\_VECTOR(123), DWG(124), DICOM\_RLE\_GRAY(125), DICOM\_RLE\_COLOR(126), DICOM\_JPEG\_GRAY(127), DICOM\_JPEG\_COLOR(128), CALS4(129), CALS2(130), CALS3(131), XWD10(132), XWD11(133), FLC(134), KDC(135), DRW(136), PLT(137), TIF\_CMP(138), TIF\_JBIG(139), TIF\_DXF(140), TIF\_UNKNOWN(141), SGI(142), SGI\_RLE(143), VECTOR\_DUMP(144), DWF(145), MPEG1(243), MPEG2(246), JPGOS(1000) (by OPTIMAL SYSTEMS encrypted JPG-file), TIFOS(1001) (by OPTIMAL SYSTEMS encrypted TIFF-file)

### Options

### Example

## GetLastError

<b>Definition</b>	GetLastError()
<b>Type</b>	Method
<b>Description</b>	Provides the error text of the last occurred error.
<b>Parameters</b>	none
<b>Return value</b>	<b>Error text</b> as string.
<b>Comment</b>	This function must be called right after the error to avoid receiving the text of a different error.
<b>Options</b>	None
<b>Example</b>	In case of errors, start an error handling routine and find the error text.

```
On error goto ErrorHandler
...
...
ErrorHandler:
ErrStr = MyAX.GetLastError()
```

## GetMainType

<b>Definition</b>	GetMainType	(long long	lObjectID, lObjectType)
<b>Type</b>	Method		
<b>Description</b>	Determines the main type of an object. In case of a multi media document type the actual main type of the document is provided.		
<b>Parameters</b>	<b>lObjectID</b>	object ID	
	<b>lObjectType</b>	object type	
<b>Return value</b>	<b>&gt; = 0</b>	main type of the object	
	<b>-22</b>	object type unknown	
	<b>-26</b>	object index unknown	
<b>Comment</b>			
<b>Options</b>			
<b>Example</b>			



**Example**

Source code example 1:

```
Helpstr=MyAX.GetObjectFields("D-Reports",131073,0)
```

**Helpstr** may contain the following string:

```
Reporttype\X\30 Annotation\X\50
```

Source code example 2:

```
Helpstr=MyAX.GetObjectFields("D-Reports",131073,1)
```

**Helpstr** may contain the following string:

```
ReportType\X\30\object1.field1 Comment\X\50\object1.field2
```



**Possible field types:**

Type	Description	Database
A	all characters without numbers	CHAR
X	all characters	CHAR
Z	only numbers	CHAR
S	male/female	CHAR
Q	yes/no	CHAR
G	Uppercase letters	CHAR
P	Patient type	CHAR
T	left/right	CHAR
L	all characters	CHAR
M	all characters	CHAR
D	Date	DATE
9	Numbers	INTEGER
I	Full text index	INTEGER
#	Decimal numbers	DECIMAL
1	Radio button	SHORT
0	Check box	SHORT
K	Button	FROM 3.50.619 !!!

Field lengths refer to the number of characters that can be entered. In case of decimal numbers, the capacity of the decimal field consists of two numbers before and after the decimal point (field length – 2).

**Possible values for catalog type:**

- 0 without catalog
- 1 list
- 2 tree
- 3 hierarchy
- 4 Database
- 5 structure tree
- 6 AddOn
- 7 Full text

**Options**

**Example**

**Source code example 1:**

```
Helpstr=MyAX.GetObjectFields("D-Reports",131073,0)
```

**Helpstr** may contain the following string:

```
Reporttype\X\30\0 Annotation\X\50\1
```

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## GetObjectName

<b>Definition</b>	GetObjectName	(long lObjectID, VARIANT* strReturnObjectName)
<b>Type</b>	Method	
<b>Description</b>	Provides the object name for the given object ID	
<b>Parameters</b>	<b>lObjectID</b>	Object ID
	<b>strReturnObjectName</b>	here the name of the object is returned
<b>Return value</b>	0 no error -1 error Error text can be determined using <b>GetLastError()</b> .	
<b>Comment</b>		
<b>Options</b>		
<b>Example</b>		

## GetObjectNameEx

<b>Definition</b>	GetObjectNameEx	(long long short VARIANT*	(lObjectID, lType, nMode, strReturnObjectName)
<b>Type</b>	Method		
<b>Description</b>	Provides the object name for the given object ID		
<b>Parameters</b>	<b>lObjectID</b>	Object ID	
	<b>lType</b>	type of the object to be determined -1 unknown type 0 object ID of a document 1 object ID of a folder 2 object ID of a register	
	<b>nMode</b>	type of the result to be returned 0 object type name 1 internal object type name 2 table name of the object type 3 UID of the object type	
	<b>strReturnObjectName</b>	here the name of the object is returned	
<b>Return value</b>	0	no error	
	-1	error	
	Error text can be determined using <b>GetLastError()</b> .		

### Comment

### Options

### Example

## GetObjectPath

<b>Definition</b>	GetObjectPath (long lObjectID, long lObjectType)
<b>Type</b>	Method
<b>Description</b>	Determines the location path of an object. Determines the folder and all registers. The object type can be a folder, register, or document.
<b>Parameters</b>	<b>lObjectID</b> object ID <b>lObjectType</b> object type
<b>Return value</b>	<b>String</b> with the object path. <b>Empty string</b> if error Error text can be determined using <b>GetLastError()</b> .

**Comment** The return string has the following format:

```
CABINETINDEX, CABINETTYP; REGISTERINDEX1, REGISTERTYPE; REGISTER  
INDEX2, REGISTERTYPE; . . . ; REGISTERINDEXn, REGISTERTYPE
```

Register information is only returned if the object is inside a register.

If the searched object is inside a register that is contained inside another register the path will look like this:  
folder ID, folder type, ID and type of the register that the object is located in, ID and type of the register that the register is located in etc.

Example:

An object is inside a register named 'REGISTER3' which itself is inside a register named REGISTER2 which again is inside a register named REGISTER1. The object path will look like this:

```
folder ID, folder type; REGISTER3 ID, REGISTER3 type; REGISTER2  
ID, REGISTER2 type; REGISTER1 ID, REGISTER1 type
```

**Options**

**Example**



## GetObjectPattern

<b>Definition</b>	GetObjectPattern	(long long long string	lIdent, lType, lMode, Titel)
<b>Type</b>	Method		
<b>Description</b>	Gets the title of the specified object as it is used or configured as a window title.		
<b>Parameters</b>	<b>lIdent</b>	Object ID	
	<b>lType</b>	object type	
	<b>lMode</b>	currently only '1'	
	<b>Titel</b>	Title of the specified object is returned	
<b>Return value</b>	<b>0</b>	no error	
	<b>-22</b>	The object is unknown	
	<b>-117</b>	Unknown fashion	

**Comment** Error text can be determined using **GetLastError()**.

### Options

### Example

```
Dim ax
Dim sRet

set ax = CreateObject("optimal_AS.Application")

ax.GetObjectPattern 846, 41, 1, sRet

set ax = Nothing
MsgBox sRet
```

## GetObjectTypeInfoImage

<b>Definition</b>	GetObjectTypeInfoImage (long long String Variant	lObjectID, lObjectType, strImageFormat, vRetImagePath)
<b>Type</b>	Method	
<b>Description</b>	Returns the icon of an object type as an image file. Currently, the file is only available in GIF format.	
<b>Parameters</b>	<b>lObjectID</b> with <b>lObjectType</b> <b>strImageFormat</b> <b>vRetImagePath</b>	ID of the object (can be 0 if it not an object an icon catalog) object type currently not used the full path to the image file is returned here
<b>Return value</b>	<b>0</b> <b>-1</b>	image file successfully transmitted image file could not be determined
<b>Comment</b>	Error texts can be determined using GetLastError().	
<b>Options</b>		
<b>Example</b>		

```
Dim a As Object
Dim vRetImagePath as Variant

Set a = CreateObject("optimal_as.application")

a.GetObjectTypeInfoImage lObjectID, lObjectType, "",
vRetImagePath
```

## GetRegTypeFromArchive

<b>Definition</b>	GetRegTypeFromArchive (String strSchrackName)
<b>Type</b>	Method
<b>Description</b>	Determines all registers with name and object type belonging to a cabinet.
<b>Parameters</b>	<b>strSchrackName</b> string with the cabinet label
<b>Return value</b>	Semicolon-separated string with registername,registertype.
<b>Comment</b>	The return string has the following format:

```
REGISTERNAME1, OBJECTTYPE1; REGISTERNAME2, OBJECTTYPE2
```

### Options

### Example

Source code example:

```
sRet=MyAX.GetRegTypeFromArchive (sArchiveName)
```

sRet may contain the following string:

```
RegisterName1, 6488064; RegisterName2, 6488065
```

## GetResultFields

<b>Definition</b>	GetResultFields	(String long long String VARIANT*	(strObjectType, IObjectType, IMode, strDelimiter, pstrFields)
<b>Type</b>	Method		
<b>Description</b>	Returns the configured fields of the current users for hit lists of the indicated type.		
<b>Parameters</b>	strObjectType	internal name of the object type (is only utilized if ID=-1 is passed)	
	IObjectType	object type	
	IMode	kind of the result to be delivered	
	strDelimiter	Separator which is used for the separation of the fieldname in the return string. If there is no separator, an ';' will be set as default.	
	pstrFields	here user configured fields are returned in this form (see remark):	
	<code>Field1; Field2; ...; Fieldn</code>		
<b>Return value</b>	0	no error	
	-7	object type unknown	
	-40	incorrect input parameter	
	Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	IMode = 0 returns display names of the fields IMode = 1 returns internal names of the fields IMode = 2 returns the OSGUIDs of the fields		
<b>Options</b>			
<b>Example</b>			

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## GetSignatureProperty

**Definition** String GetSignatureProperty (String strPropertyName)

**Type** Method

**Description** Returns a property that was previously defined with SetSignatureProperty.

**Parameters** **strPropertyName** name of the property

**Return value** Value of the property.

**Comment**

**Options**

**Example**

## GetSelectedObject

<b>Definition</b>	GetSelectedObject (Variant strSelectedObjects)														
<b>Type</b>	Method														
<b>Description</b>	Returns objects from a hit list selected with <b>SelectObject</b> .														
<b>Parameters</b>	strSelectedObjects here objects are returned in this form: <div style="border: 1px solid black; padding: 2px; display: inline-block;">ObjectID1, Objecttypel; . . . , . . . ; ObjectIDn, Objecttypen</div>														
<b>Return value</b>	<table> <tr> <td><b>3</b></td> <td>objects were selected, selection finished</td> </tr> <tr> <td><b>2</b></td> <td>selection canceled</td> </tr> <tr> <td><b>1</b></td> <td>no object selected yet</td> </tr> <tr> <td><b>0</b></td> <td>SelectObject was not executed</td> </tr> <tr> <td><b>-3</b></td> <td>cabinet unknown</td> </tr> <tr> <td><b>-5</b></td> <td>register unknown</td> </tr> <tr> <td><b>-26</b></td> <td>object ID unknown</td> </tr> </table>	<b>3</b>	objects were selected, selection finished	<b>2</b>	selection canceled	<b>1</b>	no object selected yet	<b>0</b>	SelectObject was not executed	<b>-3</b>	cabinet unknown	<b>-5</b>	register unknown	<b>-26</b>	object ID unknown
<b>3</b>	objects were selected, selection finished														
<b>2</b>	selection canceled														
<b>1</b>	no object selected yet														
<b>0</b>	SelectObject was not executed														
<b>-3</b>	cabinet unknown														
<b>-5</b>	register unknown														
<b>-26</b>	object ID unknown														
<b>Comment</b>	If this function is used in a loop, make sure that window messages are processed; otherwise no object can be selected from the hit list.														
<b>Options</b>															
<b>Example</b>															

## GetSignDocumentResult

<b>Definition</b>	GetSignDocumentResult (Variant vRetSignText)
<b>Type</b>	Method
<b>Description</b>	Returns the result of the last digital signature of a document.
<b>Parameters</b>	<b>vRetSignText</b> the chosen signature text is returned here
<b>Return value</b>	<b>0</b> signature successful <b>1</b> signature canceled <b>100</b> digital signature has not been started yet <b>101</b> digital signature still in progress <b>-1</b> error while initializing the digital signature <b>-2</b> no signature texts were configured <b>-77</b> signature model has not been initialized <b>-79</b> the file to be signed does not exist
<b>Comment</b>	Error texts can be determined using GetLastError().
<b>Options</b>	
<b>Example</b>	

```
Dim a As Object
Set a = CreateObject("optimal_as.application")

a.signdocumentex docID, docType, False, signText

abort = False
Do While abort = False
    c = a.getsigndocumentresult(signText)
    If c <> 101 Then
        abort = True
    End If
    DoEvents
Loop
```

## GetWDocPattern

<b>Definition</b>	GetWDocPattern	(long String String VARIANT* VARIANT*	(ObjectType, strPatternName, strPath, strEditorName, strFileName)
<b>Type</b>	Method		
<b>Description</b>	Copies the template file for the given object type into the specified directory		
<b>Parameters</b>	<b>ObjectType</b> <b>strPatternName</b> <b>strPath</b> <b>strEditorName</b> <b>strFileName</b>	selected object type, must be a W-Document alias name of the template destination path here the name of the editing program for this template is returned full path and name of the copied template is returned here	
<b>Return value</b>	<p>0 no error  -7 document type unknown  -35 no templates defined for this type  -36 no template found with this name  -37 template cannot be retrieved  -60 object is not a W-Document  -104 The entered alias name exists more than one time.  -105 The specified namespace was not found.  Error text can be determined using <b>GetLastError()</b>.</p>		
<b>Comment</b>	If the <i>as.cfg</i> contains the entry STOREATSERVER=1, the template is transferred by the archive server, otherwise it is copied from the template directory.		
<b>Options</b>	The name of the template can also be entered along with the intended namespace, separated by '::'.		
<b>Example</b>			

## GetWDocPatternNames

<b>Definition</b>	GetWDocPatternNames	(long String VARIANT*	IObjectType, strToken, strPatternNames)
<b>Type</b>	Method		
<b>Description</b>	Provides all alias names of the W-templates that are linked to this document type.		
<b>Parameters</b>	<b>IObjectType</b>	selected object type, must be a W-Document	
	<b>strToken</b>	separator used to separate the names in the return string	
	<b>strPatternName</b>	here the names are returned	
<b>Return value</b>	<b>0</b>	no error	
	<b>-7</b>	document type unknown	
	<b>-35</b>	no templates defined for this type	
	<b>-60</b>	object is not a W-Document	
	Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>			
<b>Options</b>			
<b>Example</b>			

## GoToDocPage

<b>Definition</b>	GoToDocPage	(long lObjectID, long lObjectType, long lPageNum)
<b>Type</b>	Method	
<b>Description</b>	Navigates in an open document to the specified page.	
<b>Parameters</b>	<b>lObjectID</b>	object index
	<b>lObjectType</b>	object type
	<b>lPageNum</b>	page number
<b>Return value</b>	0	no error
	-7	document type unknown
	-26	document index unknown
	-91	document type is not supported
	-92	document not open
	-93	document contains no pages
	Error text can be determined using <b>GetLastError()</b> .	
<b>Comment</b>	The function only supports document types with the main type <b>1 X</b> = grayscale	
	2	D = Black/White
	3	P = Color

### Options

### Example

## InfoWindow

<b>Definition</b>	InfoWindow
<b>Type</b>	Property
<b>Description</b>	Provides an object to control the info window.
<b>Parameters</b>	
<b>Return value</b>	an object to control the info window
<b>Comment</b>	In order to use the info window within an event, it is not necessary to get the object over this interface as it can be accessed in any event under the name 'InfoWindow'.

### Options

### Example

```
Dim a As Object  
  
Set a = CreateObject("optimal_as.application")  
a.InfoWindow.URL = "www.google.de"
```

## InsertFileList

<b>Definition</b>	InsertFileList	(short String long* long*	nMainType, strFileName, lReturnObjectID, lReturnObjectType)
<b>Type</b>	Method		
<b>Description</b>	Inserts one or several files (depending on main type) to the archive as new document.		
<b>Parameters</b>	<b>nMainType</b>	main type	
	<b>strFileName</b>	file with the name of the files to be inserted	
	<b>lReturnObjectID</b>	here the object index is returned	
	<b>lReturnObjectType</b>	here the object type is returned	
<b>Return value</b>	<b>0</b>	no error	
	<b>-1</b>	inserting failed	
	<b>-62</b>	currently a data sheet is open.	
	<b>-69</b>	Handoff file is empty.	
	Error exact text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	Possible main types are described in the chapter <b>Introduction</b> .		

The file names in the handoff file must be separated by **CR/LF**. The files must be located in the path determined with **GetEnvironment(0)**. (Note: If this feature, depending on computer configuration, returns short file names, paths thus must also be entered with short file names in the handoff file.)

In the enaio® client the user can define in which cabinet or folder to insert the document. Furthermore it is possible to define the document type and to insert into the filing tray. More than one file in the handoff file are allowed for the main types 1, 2 and 3 . For all other main types it must be exactly one file.

If an additional file called **FILEDATA.TXT** is indicated in the handoff file, index data for the initial indexing of the new document can be entered there. (Note: This file must be located in the working directory '**GetEnvironment(9)**' and entered as the last file after the document files in the handoff file.) In the file **FILEDATA.TXT** index fields can be specified with their name or their internal name with leading and following '%' character. Usually an indexing form is displayed to the user and he can edit the form. But if additionally the parameter **ShowNoDialog=1** is shown in this file, then no indexing file will be shown to the user in the enaio® client. I.e. the new document is inserted, but users cannot edit its index data.

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

There are two ways to insert one or more files as documents into the archive:

1. files and potential **FILEDATA.TXT** information in a string

```
[FILELIST]
#0000=C:\DOKUM~1\user\LOKALE~1\Temp\DOC0000.TIF
#0001=C:\DOKUM~1\user\LOKALE~1\Temp\DOC0001.TIF
#0002=C:\DOKUM~1\user\LOKALE~1\Temp\DOC0002.TIF
...
#000n=C:\DOKUM~1\user\LOKALE~1\Temp\DOC000n.TIF
[DATA]
FIELD0=Titel=Expose
FIELD1=Annotation=Added per script
FIELD2=InterneName_Author=Jon Doe
```

## 2. FILEDATA.TXT in the file list:

```
[FILELIST]
#0000=C:\DOKUM~1\user\LOKALE~1\Temp\DOC0000.TIF
#0001=C:\DOKUM~1\user\LOKALE~1\Temp\DOC0001.TIF
#0002=C:\DOKUM~1\user\LOKALE~1\Temp\DOC0002.TIF
...
#000n=C:\DOKUM~1\user\LOKALE~1\Temp\DOC000n.TIF
```

## Options

### Example

#### Example on the handoff file

```
C:\DOKUME~1\user\LOKALE~1\Temp\DOC000000.TIF
C:\DOKUME~1\user\LOKALE~1\Temp\OSTEMP\FILEDATA.TXT
```

#### Example of a potential FILEDATA.TXT

```
[DATA]
FIELD0=Titel=Expose
FIELD1=Annotation= Added per script
FIELD2=InterneName_Author=Jon Doe
...
ShowNoDialog=1
```

In the FILEDATA.TXT file a location can be specified now.

```
[PREDEFTARGET]
DOKUMENTINTERN
DOKUMENTTYP
SCHRANKINTERN
SCHRANK
SCHRANKIDENT
REGISTERINTERN
REGISTER
REGISTERIDENT
ABLAGE
```

Note: Internal names are read first.

- A document type must be specified.
- A cabinet type must be specified.
- A register type can be specified.
- If there is no register type, a cabinet ID must be specified.
- If there is a register type, a register ID must be specified. In this case, the cabinet ID is not necessary.
- If there is a register type, it must be located in the specified cabinet.
- The specified document type must correspond to the cabinet.
- It must be possible to create a document of the specified document type at the location (register or cabinet).

Example:

```
[PREDEFTARGET]
DOKUMENTTYP=Report
SCHRANK=Patient
REGISTER=Hospital stay
REGISTERIDENT=123456
```

If the specification in the DATA section meet all requirements, a document will be created accordingly at the specified location.

Alternatively the document can be created in the user's filing tray.

**Example:**

```
[PREDEFTARGET]
DOKUMENTTYP=Report
SCHRANK=Patient
ABLAGE=1
```

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## InsertFileListS

Corresponds to 'InsertFileList' and can be used from VB-Script.

<b>Definition</b>	InsertFileList	(short String VARIANT* VARIANT*	nMainType, strFileName, varReturnObjectID, varReturnObjectType)
-------------------	----------------	--	--

## InsertIntoArchive

<b>Definition</b>	InsertIntoArchive	(String long long	strFilename, *lpReturnObjectID, *lpReturnObjectType)
<b>Type</b>	Method		
<b>Description</b>	Allows to insert a new folder into a cabinet.		
<b>Parameters</b>	<b>strFilename</b>	handoff file (structure see comment)	
	<b>*lpReturnObjectID</b>	here the object index is returned	
	<b>*lpReturnObjectType</b>	here the object type is returned	
<b>Return value</b>	<b>0</b>	if no error	
	<b>-1</b>	inserting folder failed	
	<b>-30</b>	one or more mandatory fields not filled out	
	<b>-24</b>	field names could not be resolved	
	<b>-28</b>	invalid field value for a given field	
	<b>-31</b>	data type of entered value is not compatible with the data type of the associated field.	
	<b>-32</b>	handoff file does not exist	
	<b>-47</b>	no write permission for this object	
	<b>-64</b>	server error occurred	
	Error text can be determined using <b>GetLastError()</b> .		

**Comment** The handoff file has the following structure:

```
[EINFÜGEN]
SCHRANK=
FIELD1=
FIELD2=
...
FIELDn=
```

The following lines must be inserted in order to define values for the table controls, where the separator character equals chr(17).

```
[Table@TABLENAME]
Row0={Column1Row1 (Separator) Column2Row1}
Row1={Column1Row2 (Separator) Column2Row2}
```

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

**Options** **PFLICHTFELDER=[0;1]**  
Mandatory fields are verified for the value 1.

**VORBELEGUNG=[0;1]**  
fills in preset values fields which have not been defined explicitly.  
Default: 0

**CHECKKEYFIELDS=[0;1]**  
For the value 0 the key field check is deactivated. Default: 1

**Example** Inserting a folder into the cabinet 'Customer'

```
[EINFÜGEN]
SCHRANK=Kunde
FIELD1=Class=Customer
FIELD2=Vorlage=10.03.1997
FIELD3=optimal_AS=1
```

**Note:** Key fields are not verified here. The numbering must not contain gaps. Entries after a gap are ignored.

## InsertIntoArchiveS

Corresponds to 'InsertIntoArchive' and can be used from VB Script.

<b>Definition</b>	InsertIntoArchiveS	(String VARIANT* VARIANT*	strFilename, varReturnObjectID, varReturnObjectType)
-------------------	--------------------	---------------------------------	--

# InsertIntoDocument

<b>Definition</b>	InsertIntoDocument (String long long	strFilename, *lpReturnObjectID, *lpReturnObjectType)
<b>Type</b>	Method	
<b>Description</b>	Allows to insert a new document into a cabinet.	
<b>Parameters</b>	<b>strFilename</b>	handoff file (structure see comment)
	<b>*lpReturnObjectID</b>	here the object index is returned
	<b>*lpReturnObjectType</b>	here the object type is returned
<b>Return value</b>	<b>0</b>	if no error
	<b>-1</b>	inserting document failed
	<b>-2</b>	no folder specified
	<b>-3</b>	folder unknown
	<b>-6</b>	no document type specified
	<b>-7</b>	document type unknown
	<b>-9</b>	Document type does not belong to specified folder
	<b>-10</b>	folder identifier missing
	<b>-24</b>	field names could not be resolved
	<b>-28</b>	invalid field value for a given field
	<b>-30</b>	one or more mandatory fields not filled out
	<b>-31</b>	data type of entered value is not compatible with the data type of the associated field.
	<b>-32</b>	handoff file does not exist
	<b>-47</b>	no write permission for this object
	<b>-64</b>	server error occurred
	<b>-89</b>	The object must not be created in destination folder/register.
	Error text can be determined using <b>GetLastError()</b> .	

**Comment**

The handoff file has the following structure:

```
[EINFÜGEN]
SCHRANK=
DOKUMENT=
SCHRANK-ID=
REGISTER-ID=
MAINTYPE=
FIELD1=
FIELD2=
...
FIELDn=
DATEI1=
DATEI2=
...
DATEIn=
```

The following lines must be inserted in order to define values for the table controls, where the separator character equals chr(17).

```
[Table@TABLENAME]
Row0={Column1Row1 (Separator) Column2Row1}
Row1={Column1Row2 (Separator) Column2Row2}
```

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

**Note:** The main type of the destination document must be defined with **MAINTYPE=#** where # stands for the main type number. Possible main types are described in the chapter **Introduction**.

It is not recommended that this function is used to create links to documents which are subject to the variant administration.

## Options

**PFLICHTFELDER=[0;1]**

Mandatory fields are verified for the value 1.

**SHOWTEMPLATES=[0;1]**

The template dialog for W\_Documents is displayed if the value is 1 . The selected template is inserted. Then the entry FILE1 does not need to be inserted. If no template is selected a reference is created.

**ENABLEOPTIONS=[0;1]**

If the value is 1, the options of the template dialog are activated.

**ARCHIVIERBAR=[0;1]**

Sets the archiving status if files are specified.

**DATEILÖSCHEN=[0;1]**

Deletes the specified source files. Default is 0.

**VORBELEGUNG=[0;1]**

fills in preset values fields which have not been defined explicitly. Default: 0

**CHECKKEYFIELDS=[0;1]**

For the value 0 the key field check is deactivated. Default: 1

**FOREIGN-ID=Dokument ID**

**SYSTEM ID=**

Both of these parameters can be used to create multi-cabinet-spanning links. To do so, set the FOREIGN ID as the document ID you want to link to and the SYSTEM ID to zero and define the CABINET ID.

To attach multiple parameters, new sections have to be inserted into the file. These sections start with **MULTI** followed by internal field label of the multi-field (e.g.: MULTI\_FELD1). They are then followed by the data to be inserted in the multiple parameters, e.g.:

```
[MULTI_FELD1]
Data1=Seitennummer,Text
Data2=Seitennummer,Text
```

## Example

Inserting into the 'Customer' document, 'inbound document'.



# InsertIntoRegister

<b>Definition</b>	InsertIntoRegister	(String long long	strFilename, *lpReturnObjectID, *lpReturnObjectType)
<b>Type</b>	Method		
<b>Description</b>	Inserting a new register		
<b>Parameters</b>	<b>strFilename</b>	handoff file (structure see comment)	
	<b>*lpReturnObjectID</b>	here the object index is here	
	<b>*lpReturnObjectType</b>	here the object type is returned.	
<b>Return value</b>	<b>0</b>	if no error	
	<b>-1</b>	inserting register failed	
	<b>-2</b>	no folder specified	
	<b>-3</b>	cabinet unknown	
	<b>-4</b>	no register specified	
	<b>-5</b>	register unknown	
	<b>-8</b>	register does not belong to the folder	
	<b>-24</b>	field names could not be resolved	
	<b>-28</b>	invalid field value for a given field	
	<b>-30</b>	one or more mandatory fields not filled out	
	<b>-31</b>	data type of entered value is not compatible with the data type of the associated field.	
	<b>-32</b>	handoff file does not exist	
	<b>-47</b>	no write permission for this object	
	<b>-64</b>	server error occurred	
	<b>-89</b>	The object must not be created in destination folder/register.	

Error text can be determined using **GetLastError()**.

**Comment**

The handoff file has the following structure:

```
[EINFÜGEN]
SCHRANK=
REGISTER=
SCHRANK-ID=
REGISTER-ID=
FIELD1=
FIELD2=
...
...
FIELDn=
```

The following lines must be inserted in order to define values for the table controls, where the separator character equals chr(17).

```
[Table@TABLENAME]
Row0={Column1Row1 (Separator) Column2Row1}
Row1={Column1Row2 (Separator) Column2Row2}
```

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

HIER LOCHEN ODER DIGITAL ARCHIVIEREN



**newObjIdent** ID of the newly created object

**Return value**

- 120 Specified DMS object type is unknown.
- 121 Location for the creation cannot be determined.
- 122 A new object of the specified type cannot be created at the specified location.
- 123 Input canceled.
- 124 The necessary index write permission to create an object of the specified type is not given.
- 125 A modal dialog is opened.
- 126 Document creation in progress.
- 127 DMS object type to be created is not in the cabinet of the location.

The following return values produce a document without pages:

- 1 The necessary object write permission to save files for the specified type is not given.
- 2 At least one of the specified files does not exist.
- 3 At least one specified file does not match the document type to be created.
- 4 Too many files specified for the document type to be created.

Error text can be determined using **GetLastError()**.

**Comment**

Index data as default are optional. They can be specified as follows:

```
[Data]
Field0=field name=value
Field1 =%Field name%=value
...
```

Internal names for fields are bracketed with percent signs.

Files can be specified as:

```
[Files]
File0=C:\tmp\File1.JPG
File1=C:\tmp\File2.JPG
...
```

DeleteAfterInsert=1 deletes the files after the transfer.

Without files, the enaio® client opens the corresponding module for creating files after indexing.

OnlyIndexData=1 does not open a module but creates a document without pages. If the document type is without pages, no module is also opened.

Via ShowNoDialog=1 no index data form is opened but the object is created without user action.

**Options**

-

**Example**

-

## LicLogin

**Definition**

LicLogin (strModulName)

**Type**

Method

**Description**

Licenses the specified module.

**Parameters**

**strModulName** name of the module to be licensed

**Return value**

<b>0</b>	when the module could be licensed
<b>-1</b>	general error in function
<b>&gt;0</b>	license error number of application server

**Comment** If the specified module has already been licensed for the current workstation, this license will be used.

**Options**

**Example**

## LicLogout

**Definition** LicLogout (strModulName)

**Type** Method

**Description** Unblocks the license for the specified module.

**Parameters** **strModulName** name of the module to be unblocked

**Return value**

<b>0</b>	if the license could be unblocked
<b>-1</b>	if the license could not be unblocked

**Comment** Only modules previously licensed with LicLogin can be unblocked with LicLogout

**Options**

**Example**

## LinkDocuments

<b>Definition</b>	LinkDocuments	(long long long long	lObjectID1, lObjectType1, lObjectID2, lObjectType2)
<b>Type</b>	Method		
<b>Description</b>	Creates a link between two documents.		
<b>Parameters</b>	<b>lObjectID1</b>	ID of the first document	
	<b>lObjectType1</b>	type of the first document	
	<b>lObjectID2</b>	ID of the second document	
	<b>lObjectType2</b>	type of the second document	
<b>Return value</b>	<b>0</b>	no error	
	<b>-20</b>	one of the object types is not a document type, the erroneous type can be found in the error text	
	<b>-29</b>	one or both object IDs are unknown	
	<b>-34</b>	this link already exists	
	<b>-86</b>	object indexes are identical	
	<b>-87</b>	objects of the type portfolio are not allowed	
	<b>-88</b>	objects from the filing tray are not allowed	
		Error text can be determined using <b>GetLastError()</b> .	
<b>Comment</b>	The link between two documents is saved in their notes.		

### Options

### Example

## MergeArchives

**Definition** MergeArchives (long ISourceID,  
long ITargetID,  
long IObjectType)

**Type** Method

**Description** Combines the contents of two folders.

**Parameters** ISourceID source folder  
ITargetID destination folder  
IObjectType object type

**Return value** 0 no error  
-22 object type unknown  
-47 insufficient rights to create objects  
-80 source folder unknown  
-81 destination folder unknown  
-82 updating database failed  
Error text can be determined using **GetLastError()**.

**Comment**

**Options**

**Example**

## MoveObject

<b>Definition</b>	MoveObject	(long long long long	lObjectID, lObjectType, lFolderID, lRegisterID)
<b>Type</b>	Method		
<b>Description</b>	Moves an object (register or document).		
<b>Parameters</b>	<b>lObjectID</b>	ID of the object to be moved	
	<b>lObjectType</b>	corresponding object type	
	<b>lFolderID</b>	folder ID of the destination folder	
	<b>lRegisterID</b>	register ID of the destination folder	
<b>Return value</b>	<b>0</b>	no error	
	<b>-5</b>	register type unknown (register ID belongs to wrong object type)	
	<b>-7</b>	document type unknown	
	<b>-13</b>	register ID unknown	
	<b>-14</b>	folder ID unknown	
	<b>-29</b>	object ID invalid	
	<b>-68</b>	Folder cannot be moved.	
	<b>-82</b>	updating database failed	
	<b>-90</b>	The specified object is a reference copy	
	<b>539</b>	Violation of object type relation	
	Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	If a register ID > 0 is specified, the folder ID is ignored. If register ID and folder ID = 0, the object is moved to the root.		
	Reference documents cannot be moved.		
<b>Options</b>			
<b>Example</b>			

## OleDdeRequest

<b>Definition</b>	OleDdeRequest	(String String	strFunctionName, strParameter)
<b>Type</b>	Method		
<b>Description</b>	This is a legacy function and is no longer supported.		
<b>Parameters</b>	<b>strFunctionName</b>	Name of the function	
	<b>strParameter</b>	Parameter as string	
<b>Return value</b>			
<b>Comment</b>	COM functions must be called directly instead of using this function.		
<b>Options</b>			
<b>Example</b>			

## OpenAboDialog

<b>Definition</b>	OpenAboDialog	(long long long long String	IHwnd, IObjectID, IObjectType, IFlags, strInfoText)
<b>Type</b>	Method		
<b>Description</b>	Displays the subscription dialog.		
<b>Parameters</b>	<b>IHwnd</b>	window handle	
	<b>IObjectID</b>	document ID	
	<b>IObjectType</b>	document type	
	<b>IFlags</b>	see comment	
	<b>strInfoText</b>	info text that is inserted into the dialog's info field	
<b>Return value</b>	<b>1</b>	dialog canceled by user	
	<b>0</b>	no error	
	<b>-7</b>	object type unknown	
	<b>-26</b>	IObjectID is unknown	
	<b>-62</b>	A dialog is already open.	
	Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	By setting flags certain check boxes can be set in the dialog. These flags can be combined.		
	<b>1</b>	check box document changed	
	<b>2</b>	check box index data changed	
	<b>4</b>	check box object created	
	<b>8</b>	check box object deleted	
<b>Options</b>			
<b>Example</b>			

## OpenDataDlg

<b>Definition</b>	OpenDataDlg	(long long short	lObjectID, lObjectType, nMode)
<b>Type</b>	Method		
<b>Description</b>	Opens the data sheet of the specified object.		
<b>Parameters</b>	<b>lObjectID</b>	object index	
	<b>lObjectType</b>	object type	
	<b>nMode</b>	editing mode (see comment)	
<b>Return value</b>	<b>0</b>	no error	
	<b>-7</b>	object unknown	
	<b>-11</b>	invalid document identifier (= 0)	
	Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	Displayed by the enaio® client. Possible values for nMode are:		
	<b>0</b>	the data sheet is displayed read-only and waiting for input	
	<b>1</b>	the data sheet is opened for editing and waiting for input	
	<b>2</b>	the data sheet is displayed read-only and not waiting for input	
	<b>3</b>	the data sheet is opened for editing and not waiting for input	

### Options

### Example

## OpenObjectID

<b>Definition</b>	OpenObjectID (long long short	lObjectID, lObjectType, nMode)
<b>Type</b>	Method	
<b>Description</b>	<p>Opens the specified object.</p> <p>If the object is a <b>document</b>, then the image is displayed and/or the corresponding document is loaded (W-Document). If the document has no images, the data sheet is opened.</p> <p>If the object is a <b>folder</b> or <b>register</b> then the corresponding folder is opened.</p>	
<b>Parameters</b>	<b>lObjectID</b> <b>lObjectType</b> <b>nMode</b>	index of the object to be opened object type Sets for W-Documents (and only those) if the document will be opened as read-only or for editing. If the parameter is <b>0</b> , the document will be opened as usual as read-only. If the mode = <b>1</b> , the document will be opened for editing. nMode = <b>2</b> , If there are multiple document variants, instead of the active variant the document with the specified ID will open in read-only mode. nMode = <b>3</b> , If there are multiple document variants, instead of the active variant the document with the specified ID will open for editing. read-only mode.
<b>Return value</b>	<b>0</b> <b>-11</b>	no error if lObjectID = 0 Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	Displayed by the enaio® client.	
<b>Options</b>		
<b>Example</b>	A document will be opened as read-only: <pre style="border: 1px solid black; padding: 2px;">intRet = MyAX.OpenObjectID(lngID, lngType, 0)</pre>	

## OpenResultList

<b>Definition</b>	OpenResultList	(String String	strIDList, strTitle)
<b>Type</b>	Method		
<b>Description</b>	Opens a hit list with the passed objects.		
<b>Parameters</b>	<b>strIDList</b>	object list in this form: ID,Objecttype;ID1,Objecttype1...	
	<b>strTitle</b>	window title	
<b>Return value</b>	0	no error	
	-1	hit list could not be created	

### Comment

### Options

### Example

```
Dim a As Object
Set a = CreateObject("optimal_as.application")
a.OpenResultList "873722,131089", "test"
```

## OpenObjectIDEx

<b>Definition</b>	OpenObjectIDEx	(long lObjectID, long lObjectType, BOOL bWriteProtected, BOOL bActivateIfOpen)
<b>Type</b>	Method	
<b>Description</b>	Opens the specified object. If the object is a document, the image is displayed an/or the corresponding document is loaded (W-Document). If the document has no images, the data sheet is opened. If the object is a cabinet or register then the corresponding folder is opened.	
<b>Parameters</b>	<b>lObjectID</b>	object ID
	<b>lObjectType</b>	object type
	<b>bWriteProtected</b>	TRUE, if the document will be opened as read-only
	<b>bActivateIfOpen</b>	TRUE, if an already open window of this type will be activated
<b>Return value</b>	0	no error
	-11	object ID invalid
	Error text can be determined using <b>GetLastError()</b> .	
<b>Comment</b>		
<b>Options</b>		
<b>Example</b>		

## OpenURL

<b>Definition</b>	OpenURL	(String BOOL BOOL	strURL, bShowAddressBar, bOpenNewWindow)
<b>Type</b>	Method		
<b>Description</b>	Opens an URL.		
<b>Parameters</b>	<b>strURL</b> <b>bShowAddressBar</b> <b>bOpenNewWindow</b>	the address to be opened <b>FALSE</b> if the address bar is to be hidden <b>TRUE</b> if the address is to be opened in a new window	
<b>Return value</b>	<b>0</b> <b>-1</b>	no error URL-window could not be opened	
<b>Comment</b>			
<b>Options</b>			
<b>Example</b>			

## OpenWorkItem

<b>Definition</b>	OpenWorkItem (String strWorkItemID)
<b>Type</b>	Method
<b>Description</b>	Opens a process step in the inbox of the logged-on user.
<b>Parameters</b>	strWorkItemID      Id of the process step to be opened.
<b>Return value</b>	<b>0</b> no error <b>-113</b> workflow module is not available <b>-114</b> Id of the process step not found in the inbox <b>-115</b> internal error while opening the process step Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	This functions exclusively opens process steps which are located in the inbox of the currently logged-on user.
<b>Options</b>	
<b>Example</b>	

## PrintDocumentID

<b>Definition</b>	PrintDocumentID (String strParam)
<b>Type</b>	Method
<b>Description</b>	Prints the given objects if the objects are of the type 'document'. Except for documents of the W type.
<b>Parameters</b>	<b>strParam</b> string with the following content: <div style="border: 1px solid black; padding: 2px;">INDEX, OBJECTTYPE</div>
<b>Return value</b>	<p>0 no error</p> <p>-1 not a document type object</p> <p>-7 document type unknown</p> <p>-40 handover parameter wrong</p> <p>Error text can be determined using <b>GetLastError()</b>.</p>
<b>Comment</b>	<p>The print dialog of enaio® client will open.</p> <p>More than one object can be specified. The parameter string has the following form:</p> <div style="border: 1px solid black; padding: 2px;">INDEX1, OBJECTTYPE ... INDEXn, OBJECTTYPE</div> <p>The object information are separated with spaces. Please note that only one object type can be used.</p> <p>The file name can also be passed. The file would have the following structure:*</p> <div style="border: 1px solid black; padding: 2px;"> <pre>INDEX1, OBJECTTYPE\r\n ... INDEXn, OBJECTTYPE\r\n</pre> </div> <p>* <b>\r\n</b> stands for a line break, <b>CR</b> and <b>LF</b>.</p>
<b>Options</b>	
<b>Example</b>	

## RefreshFolderWindow

<b>Definition</b>	RefreshFolderWindow(long long short	lObjectID, lObjectType, lObjectID2Select)
<b>Type</b>	Method	
<b>Description</b>	Updates all folder and register windows open in the client and specified by ID and type. If this window is the active one, a new object can be selected by specifying ID2Select.	
<b>Parameters</b>	<b>lObjectID</b> <b>lObjectType</b> <b>lObjectID2Select</b>	object index object type object which will be selected ( <b>0</b> if you want to keep the previous selection)
<b>Return value</b>		
<b>Comment</b>	<p>This function updates all open folder windows of the specified folder/register. A new selection on the object handed off to the function will only take effect if the window is the client's active window.</p> <p>Window updates may take a while, depending on the system. During this time the client will be temporarily unavailable. For that reason, consider carefully when to carry out this function.</p>	

### — Options

### Example

## ScanDocument

<b>Definition</b>	ScanDocument	(long long short	(long long short	lObjectID, lObjectType, nMode)
<b>Type</b>	Method			
<b>Description</b>	Allows images to be attached to an existing document.			
<b>Parameters</b>	<b>lObjectID</b>	object index		
	<b>lObjectType</b>	object type		
	<b>nMode</b>	mode (see comment)		
<b>Return value</b>	0	no error		
	-11	invalid document identifier		
	-19	object is not assigned to any folder		
	-20	specified object type is not a document type		
	-21	document already archived		
	Error text can be determined using <b>GetLastError()</b> .			
<b>Comment</b>	If nMode has the value 1, the index form is opened before the scan window. If this form is left using 'Cancel', the scan window will not be opened.			
	The scan dialog of the client will be used.			
	This function cannot be used to change already archived documents.			

### Options

### Example

# ShowVariantsDialog

**Definition** ShowVariantsDialog(long lObjectID, long lObjectType, BOOL bCreateSingleVariant, BOOL bAllowDragDrop, BOOL bOpenAfterCreation, BOOL bSetNewVersionActive, long lHwnd, VARIANT vRetNewObjectID)

**Type** Method

**Description** Displays the variant administration dialog.

**Parameters**

<b>lObjectID</b>	document ID
<b>lObjectType</b>	document type
<b>bCreateSingleVariant</b>	specifies whether the user can create only one variant
<b>bAllowDragDrop</b>	specifies whether dragging and dropping
<b>bOpenAfterCreation</b>	is allowed in this dialog specifies whether a new variant is opened
<b>bSetNewVersionActive</b>	directly after creation specifies whether a new variant is marked
<b>lHwnd</b>	as active window handle
<b>vRetNewObjectID</b>	returns IDs of new created variants separated with semicolons

**Return value**

1	dialog canceled by user
0	no error
-7	object type unknown
-26	lObjectID is unknown
-60	object not of type W-Document
-62	A dialog is already open.
-93	object contains no pages

Error text can be determined using **GetLastError()**.

**Comment**

**Options**

**Example**

## SelectObject

<b>Definition</b>	SelectObject	(long long BOOL String	(long long BOOL String	lObjectID, lObjectType, bMultiSelect strTitle)
<b>Type</b>	Method			
<b>Description</b>	Opens the hit list for object selection.			
<b>Parameters</b>	<b>lObjectID</b>	folder or register ID		
	<b>lObjectType</b>	object type of the folder or register		
	<b>bMultiSelect</b>	<b>TRUE</b> for multiple selection		
	<b>strTitle</b>	window title		
<b>Return value</b>	<b>1</b>	selection was started		
	<b>-25</b>	object is not a folder or register		
	<b>-26</b>	Object ID invalid		
	<b>-27</b>	a hit list is already opened for this object		
	Error text can be determined using <b>GetLastError()</b> .			
<b>Comment</b>	The open hit list has two new buttons in the toolbar to select the selected objects or to cancel the selection.			
<b>Options</b>				
<b>Example</b>				

## SendMail

<b>Definition</b>	SendMail	(String short	strFileList, nDelete)
<b>Type</b>	Method		
<b>Description</b>	Opens a form in order to send an e-mail.		
<b>Parameters</b>	<b>strFileList</b>	file list that is attached to the e-mail	
	<b>nDelete</b>	0, if the files must not be deleted, else 1	
<b>Return value</b>			
<b>Comment</b>			
<b>Options</b>			
<b>Example</b>			



## SetPlannedRetention

<b>Definition</b>	SetPlannedRetention (long long String	lObjectID, lObjectType, strRetentionDate)
<b>Type</b>	Method	
<b>Description</b>	Sets the scheduled retention date for a document which will be archived.	
<b>Parameters</b>	<b>lObjectID</b> <b>lObjectType</b> <b>strRetentionDate</b>	object ID object type retention date with the format DD.MM.YYYY
<b>Return value</b>	<b>0</b> retention date successfully set <b>-1</b> retention date could not be set <b>-7</b> object type unknown <b>-22</b> an object with the specified ID does not exist <b>-38</b> invalid document type (a folder or register type was specified) <b>-40</b> invalid parameter passed	
<b>Comment</b>	Error texts can be determined using GetLastError().	

### Options

### Example

```
Dim a As Object

Set a = CreateObject("optimal_as.application")

a.SetPlannedRetention lObjectID, lObjectType, "12.10.2010"
```

## SetResultListSelection

<b>Definition</b>	SetResultListSelection (String strObjectIDs)
<b>Type</b>	Method
<b>Description</b>	Selects objects in the active hit list.
<b>Parameters</b>	<b>strObjectIDs</b> IDs of the objects to be selected, separated by commas
<b>Return value</b>	Number of selected objects
<b>Comment</b>	The selection is only performed in the currently active window.
<b>Options</b>	
<b>Example</b>	



## SetWaitingCursor

<b>Definition</b>	SetWaitingCursor (short sShow)
<b>Type</b>	Method
<b>Description</b>	Shows the mouse pointer as an hourglass.
<b>Parameters</b>	<b>sShow</b> Turn on hourglass (1) and off again (0).
<b>Return value</b>	<b>No return value</b>
<b>Comment</b>	Must be turned off at the end of the script.
<b>Options</b>	-

**Example**

```
SetWaitingCursor(1)

Dim dteWait

dteWait = DateAdd("s", 10, Now())

Do Until (Now() > dteWait)
Loop

SetWaitingCursor(0)

ResultCode=1
WriteToFile
```

## SignDocument

<b>Definition</b>	SignDocument (long long      IObjectID, IObjectType)
<b>Type</b>	Method
<b>Description</b>	Opens the digital signature dialog of a document.
<b>Parameters</b>	<b>IObjectID</b> ID of the document to be signed <b>IObjectType</b> type of the document to be signed
<b>Return value</b>	<b>No return value</b>
<b>Comment</b>	The function calls the signature in an own thread. To obtain the return value the function GetSignDocumentResult can be used.
<b>Options</b>	-

### Example

```
Dim a As Object
Set a = CreateObject("optimal_as.application")

a.signdocument docID, docType

abort = False
Do While abort = False
    c = a.getsigndocumentresult()
    If c <> 101 Then
        abort = True
    End If
    DoEvents
Loop
```

# SignDocumentEx

<b>Definition</b>	SignDocumentEx	(long long BOOL Variant	(long long BOOL Variant	lObjectID, lObjectType, bWaitForCompletion, vRetSignText)
<b>Type</b>	Method			
<b>Description</b>	Opens the digital signature dialog of a document.			
<b>Parameters</b>	<b>lObjectID</b>	ID of the document to be signed		
	<b>lObjectType</b>	type of the document to be signed		
	<b>bWaitForCompletion</b>	specifies whether the function will wait for the signature to be completed.		
	<b>vRetSignText</b>	the chosen signature text is returned here		
<b>Return value</b>	0 signature successful, launched successfully See <b>GetSignDocumentResult</b>			
<b>Comment</b>	The function calls the signature in an own thread. To obtain the return value the function GetSignDocumentResult can be used.			

**Options**

**Example**

```

Dim a As Object
Set a = CreateObject("optimal_as.application")

a.signdocumentex docID, docType, False, signText

abort = False
Do While abort = False
    c = a.getsigndocumentresult(signText)
    If c <> 101 Then
        abort = True
    End If
    DoEvents
Loop
    
```

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

## StartArchiveRequest

<b>Definition</b>	StartArchiveRequest (String strFileName)
<b>Type</b>	Starts a cabinet query
<b>Description</b>	Provides the error text of the last occurred error.
<b>Parameters</b>	<b>strFileName</b> name of the query file
<b>Return value</b>	<b>file name</b> or <b>READY</b> , if QUERYWINDOW <> 0. <b>Empty string</b> , if an error occurred. Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	The query file must have the following structure:

```
[ANFRAGE]
CABINET =
CLAUSE1=
CLAUSE2=
...
...
CLAUSEn=
DATENFELDER=0 (1)
DATENHEADER=0 (1)
ANFRAGEFENSTER=0 (1, 2)
AUTOSTERN=0 (1, 2)
VOLLTEXT=
```

For DATABASEFIELDS=1 additionally a section called: [QUERYFIELDS] can be defined. It allows you to specify all fields to be displayed in the hit list of the file; thus only these fields will be queried.

### Example:

The document type 'Krankendossier' has 3 fields: **form type**, **topic**, and **comment**. If only the values for **topic** and **comment** are meant to be queried, the section [QUERYFIELDS] would look like this:

```
[QUERYFIELDS]
Field0=topic
Field1=comment
```

The numbering of the fields must be continuous! This functionality also applies to the functions **StartRegRequest** and **StartDocRequest**.

If line breaks are contained in the returned field values, they are replaced with ASCII-character no. 17; so the programmer can decide whether to reinsert the line breaks when outputting the values in order to assure a proper output. The return file has the following structure:

```
INDEX1, OBJECTTYPE
INDEX2, OBJECTTYPE
...
...
INDEXn, OBJECTTYPE
```

An empty file is a valid result.

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

**Options****DATAFIELDS:**

If this option is switched on with DATAFIELDS=1, in addition to index and object type all object data is written into the file. The single columns are separated with the special character <TAB>.

A line in the result table would look like this:

```
1234,131071<TAB>Heiner<TAB>Lauterbach<TAB>Actor<CR><LF>
```

**DATAHEADER:**

If this option is switched on with DATAHEADER=1, then the first line of the result file will contain the column headings. The first line in the result table would look like this:

```
OSID,OSTYPE<TAB>First name<TAB>Last
name<TAB>Occupation<CR><LF>
```

**QUERYWINDOW:**

If a value for query window was specified, the following actions can be caused:

- 1 The query window will open if a query result is available. If no query result is available, nothing happens.
- 2 The query window will open if a query result is available. If no query result is available, a message will be displayed informing the user that the query did not return any results.

**AUTOASTERISK:**

Specifies automatic adding of asterisks to query values.

Possible values:

- 0 autoasterisk setting as in the client
- 1 autoasterisk active
- 2 switch off autoasterisk
- 3 autoasterisk at the end
- 33 autoasterisk at the front
- 35 autoasterisk at the front and end

**Example****Query file for a query in the cabinet Customer**

```
[ANFRAGE]
SCHRANK=Kunde
CLAUSE1=Customer@Class=Customer
CLAUSE2=Customer@Vorlage=10.03.1997
CLAUSE3=Customer@optimal_AS=1
DATAHEADER=1
DATAFIELDS=1
```

**Note:** Clauses must be numbered sequentially. If there is a sequence gap, the following logical expressions will be ignored. Cabinet and field identifiers must exactly match the identifiers in the object definition.

The request data is not verified. SQL errors may occur if e.g. a date is expected and text was specified.

# StartDocRequest

<b>Definition</b>	StartDocRequest (String strFileName)
<b>Type</b>	Method
<b>Description</b>	Starts a document query.
<b>Parameters</b>	<b>strFileName</b> name of the query file
<b>Return value</b>	<i>File name</i> or <b>READY</b> , if QUERYWINDOW <> 0 Empty string, if an error occurred. Error text can be determined using <b>GetLastError()</b> .

**Comment** The query file must have the following structure:

```
[ANFRAGE]
SCHRANK=
DOKUMENT=
CLAUSE1=
CLAUSE2=
...
...
CLAUSEn=
DATENFELDER=0 (1)
DATENHEADER=0 (1)
ANFRAGEFENSTER=0 (1, 2)
AUTOSTERN=0 (1, 2)
VOLLEXT=
```

In order to create a register hit list, the register name must be specified. See also 'StartRegRequest'

The return file has the following structure:

```
INDEX1, OBJECTTYPE
INDEX2, OBJECTTYPE
...
...
INDEXn, OBJECTTYPE
```

**An empty file is a valid result.**

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

**Options**

**DATAFIELDS:**

If this option is switched on with DATAFIELDS=1, in addition to index and object type all object data is written into the file. The single columns are separated with the special character <TAB>. A line in the result table would look like this:

```
1234, 131071<TAB>Heiner<TAB>Lauterbach<TAB>Actor<CR><LF>
```

**DATAHEADER:**

If this option is switched on with DATAHEADER=1, then the first line of the result file will contain the column headings. The first line in the result table would look like this:

```
OSID, OSTYPE<TAB>First name<TAB>Last name<TAB>Occupation<CR><LF>
```

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

**QUERYWINDOW:**

If a value for query window was specified, the following actions can be caused:

- 1 The query window will open if a query result is available. If no query result is available, nothing happens.
- 2 The query window will open if a query result is available. If no query result is available a message will be displayed informing, that the query did not return any results.

**AUTOASTERISK:**

Specifies automatic adding of asterisks to query values.

- 0 autoasterisk setting as in the client
- 1 autoasterisk active
- 2 switch off autoasterisk
- 3 autoasterisk at the end
- 33 autoasterisk at the front
- 35 autoasterisk at the front and end

**TYPE=2(0,1)**

Specifies, what type of hit list will be created.

- 0 folder hit list
- 1 register hit list
- 2 document hit list

If the type is not specified, a document hit list is generated.

**LOCALSEARCH=0,1,2**

Specifies, if documents without register reference will be included in the query. If this value is 2, the user setting is applied to the client application.

Furthermore it can be specified in as.cfg, if this value (the setting in the client) will be automatically included in the query file. To do so the following needs to be added in the CLIENT section:

```
AUTOLOCALSEARCH=1
```

**Example**

Query file for a query for the document type **InboundDocument** in the cabinet **Customer**

```
[ANFRAGE]
SCHRANK=Kunde
REGISTER=Register
DOKUMENT=InboundDocument
CLAUSE1=Customer@Name=Müller
CLAUSE2= Register@Type=job
CLAUSE3=Eingangsbeleg@Vorlage=10.03.1997
DATAHEADER=1
DATAFIELDS=1
```

**Note:** Clauses must be numbered sequentially. If there is a sequence gap, the following logical expressions will be ignored.

Cabinet and field identifiers must exactly match the identifiers in the object definition.

The request data is not verified. SQL errors may occur if e.g. a date is expected and text was specified.

**See also:** StartArchiveRequest

## StartRegRequest

<b>Definition</b>	StartRegRequest (String strFileName)
<b>Type</b>	Method
<b>Description</b>	Starts a register query.
<b>Parameters</b>	<b>strFileName</b> name of the query file
<b>Return value</b>	<b>File name</b> or <b>READY</b> , if QUERYWINDOW <> 0 <b>Empty string</b> , if an error occurred. Error text can be determined using <b>GetLastError()</b> .
<b>Comment</b>	The query file must have the following structure:

```
[ANFRAGE]
SCHRANK=
REGISTER=
CLAUSE1=
CLAUSE2=
...
...
.CLAUSEn=
DATENFELDER=0 (1)
DATENHEADER=0 (1)
ANFRAGEFENSTER=0 (1, 2)
AUTOSTERN=0 (1, 2)
VOLLTEXT=
```

The return file has the following structure:

```
INDEX1, OBJECTTYPE
INDEX2, OBJECTTYPE
.
.
.
INDEXn, OBJECTTYPE
```

An empty file is a valid result.

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

### Options

#### DATAFIELDS:

If this option is switched on with DATAFIELDS=1, in addition to index and object type all object data is written into the file. The single columns are separated with the special character <TAB>. A line in the result table would look like this:

```
1234, 131071<TAB>Heiner<TAB>Lauterbach<TAB>Actor<CR><LF>
```

#### DATAHEADER:

If this option is switched on with DATAHEADER=1, then the first line of the result file will contain the column headings. The first line in the result table would look like this:

```
OSID, OSTYPE<TAB>First name<TAB>Last
name<TAB>Occupation<CR><LF>
```

#### QUERYWINDOW:

If a value for query window was specified, the following actions can be caused:

- 1 The query window will open if a query result is available. If no query result is available, nothing happens.
- 2 The query window will open if a query result is available. If no query result is available, a message will be displayed informing the user that the query did not return any results.

**AUTOASTERISK:**

Specifies automatic adding of asterisks to query values.

- 0 autoasterisk setting as in the client
- 1 autoasterisk active
- 2 switch off autoasterisk
- 3 autoasterisk at the end
- 33 autoasterisk at the front
- 35 autoasterisk at the front and end

**Example**

Query file for a query in the 'Customer' register **Register**

```
[ANFRAGE]
SCHRANK=Kunde
REGISTER=Register
CLAUSE1=Customer@Name=Müller
CLAUSE2=Register@Typ=Auftrag
CLAUSE3=Customer@optimal_AS=1
DATAHEADER=1
DATAFIELDS=1
```

**Note:** Clauses must be numbered sequentially. If there is a sequence gap, the following logical expressions will be ignored. Cabinet and field identifiers must exactly match the identifiers in the object definition.

The request data is not verified. SQL errors may occur if e.g. a date is expected and text was specified.

**See also:** [StartArchiveRequest](#)

## StartClientRequest

<b>Definition</b>	StartClientRequest (string request)
<b>Type</b>	Method
<b>Description</b>	Starts the formulated client request. Full-text hitlists are displayed with facets.
<b>Parameters</b>	<b>request</b> Request in JSON format
<b>Comment</b>	Description JSON <b>request</b> start object <b>objecttype</b> UINT, object type of the request <b>fulltext</b> string, full-text query string <b>fields</b> array, JSON elements of type field <b>field</b> consists of: <b>iname</b> String, internal name of the field to be queried <b>value</b> String, value of the field to be queried
<b>Return value</b>	<b>0</b> Request executed <b>-125</b> A modal dialog is opened. <b>-140</b> JSON has no member 'request' <b>-141</b> Member 'request' has no member 'objecttype' <b>-143</b> Member 'objecttype' has unexpected data type <b>-144</b> Member 'objecttype' is invalid <b>-145</b> Specified object type is unknown <b>-146</b> Specified object type is not in the full-text index <b>-147</b> Member 'fulltext' has unexpected data type <b>-148</b> Member 'fulltext' has no value <b>-149</b> JSON contains parse errors <b>-150</b> Member 'fields' is not an array <b>-151</b> Array member 'iname' or 'value' missing <b>-152</b> Array member 'iname' or 'value' has no value <b>-153</b> Specified field not found

Error exact text can be determined using `GetLastError()`.

### Example

```
Dim ax
Dim ret
Dim req
set ax = CreateObject("optimal_AS.Application")

'10.1.4.159#4000
req = "{ \"request\": { \"objecttype\": 393226, \"fulltext\": \"optimal\", \"fields\": [ { \"iname\": \"MAIL_FROM\", \"value\": \"multi*\" } ] } }"
ret = ax.StartClientRequest(req)

if ret <> 0 then
    MsgBox ax.GetLastError() & " (" & ret & ")"
else
    ax.ActivateApp 1
end if

set ax = nothing
Application = nothing
```

## StoreNotice

<b>Definition</b>	StoreNotice	(long long String	lObjectID, lObjectType, strFileName)
<b>Type</b>	Method		
<b>Description</b>	Saves a note for an existing object (folder, register, document).		
<b>Parameters</b>	<b>lObjectID</b>	object ID	
	<b>lObjectType</b>	object type	
	<b>strFileName</b>	file name of the note with 'txt' as file extension	

<b>Return value</b>	<b>0</b>	no error
	<b>-1</b>	no destination object specified
	<b>-3</b>	cabinet unknown
	<b>-7</b>	document type unknown
	<b>-14</b>	folder identifier unknown
	<b>-15</b>	document identifier unknown
	<b>-32</b>	note file does not exist
	<b>-41</b>	file name empty
	Error text can be determined using <b>GetLastError()</b> .	

**Comment**

The note must be provided as an ASCII text file with 'txt' as file extension. Independently of the success or failure of the function, the given file will not be deleted by the archiving system. If it will be no longer required after the function call, it is recommended to have it deleted by the calling application.

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break. If, instead of a file, a string is passed and notes are stored as files on the server, the client itself creates the file.

### Options

### Example

## TransformXML

<b>Definition</b>	TransformXML	(String String VARIANT	strXMLFile, strXSLFile, varRetString)
<b>Type</b>	Method		
<b>Description</b>	Converts an XML file using the XSL file into the specified format.		
<b>Parameters</b>	<b>strXMLFile</b>	full path and file name of the XML file	
	<b>strXSLFile</b>	full path and file name of the XSL file	
	<b>varRetString</b>	the converted object is here returned as a string.	
<b>Return value</b>	<b>0</b>	no error	
	<b>-32</b>	one of the two source files does not exist	
	<b>-100</b>	files could not be converted	
	<b>-101</b>	XML file could not be loaded	
	<b>-102</b>	XSL file could not be loaded	
	<b>-103</b>	XML object could not be created	
	Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>			
<b>Options</b>			
<b>Example</b>			

## UndoCheckOut

<b>Definition</b>	UndoCheckOut	(long long	IDocID, IDocType)
<b>Type</b>	Method		
<b>Description</b>	Undoes the check-out procedure.		
<b>Parameters</b>	<b>IDocID</b> <b>IDocType</b>	Document ID document type	
<b>Return value</b>	<b>0</b> no error <b>-1</b> undo failed <b>-7</b> unknown document type <b>-53</b> document was not checked out <b>-56</b> document name could not be determined in cache Error text can be determined using <b>GetLastError()</b> .		
<b>Comment</b>	Only for documents that the logged-in user has checked out themselves.		
<b>Options</b>			
<b>Example</b>			

## UnlinkDocuments

<b>Definition</b>	UnlinkDocuments	(long lObjectID1, long lObjectType1, long lObjectID2, long lObjectType2 )
<b>Type</b>	Method	
<b>Description</b>	Deletes a link between two documents.	
<b>Parameters</b>	<b>lObjectID1</b>	ID of the first document
	<b>lObjectType1</b>	type of the first document
	<b>lObjectID2</b>	ID of the second document
	<b>lObjectType2</b>	type of the second document
<b>Return value</b>	<b>0</b>	no error
	<b>-29</b>	one or both object IDs are unknown
	<b>-86</b>	object indexes are identical
	<b>-88</b>	objects from the filing tray are not allowed
	Error text can be determined using <b>GetLastError()</b> .	
<b>Comment</b>	This function is available from 4.00 SPII. The link between two documents is saved in their notes.	
<b>Options</b>		
<b>Example</b>		

# UpdateArchiveData

<b>Definition</b>	UpdateArchiveData (String long long	strFilename, *lpReturnObjectID, *lpReturnObjectType)
<b>Type</b>	Method	
<b>Description</b>	Performs an update of the folder data.	
<b>Parameters</b>	<b>strFilename</b>	handoff file (structure see comment)
	<b>*lpReturnObjectID</b>	here the object index is returned after successful insertion
	<b>*lpReturnObjectType</b>	here the object type is returned after successful insertion
<b>Return value</b>	<b>0</b>	if no error
	<b>-3</b>	cabinet unknown
	<b>-10</b>	folder identifier missing
	<b>-14</b>	folder identifier unknown
	<b>-18</b>	update failed
	<b>-24</b>	field names could not be resolved
	<b>-28</b>	invalid field value for a given field
	<b>-30</b>	one or more mandatory fields not filled out
	<b>-31</b>	data type of entered value is not compatible with the data type of the associated field.
	<b>-32</b>	handoff file does not exist
	<b>-47</b>	no write permission for this object
	<b>-64</b>	server error occurred
	Error text can be determined using <b>GetLastError()</b> .	

**Comment** The handoff file has the following structure:

```
[AKTUALISIEREN]
SCHRANK=
SCHRANK-ID=
FIELD1=
FIELD2=
...
...
FIELDn=
Mode=1
```

If mode = 1, all non-specified fields are not reset. See annotation 'WARNING'

The following lines must be inserted in order to update the table controls, where the separator character equals chr(17).

```
[Table@TABLENAME]
Mode=0[1] 0 = Tabelle leeren, 1 = Tabelle anhängen
Row0={Column1Row1 (Separator) Column2Row1}
Row1={Column1Row2 (Separator) Column2Row2}
```

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

**Options** **PFLICHTFELDER=[0;1]**  
Mandatory fields are verified for the value 1.

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

**Example**

## Updating the 'Customer' folder

```
[AKTUALISIEREN]
SCHRANK=Kunde
SCHRANK-ID=4711
FIELD1=Class=Customer
FIELD2=Vorlage=10.03.1997
FIELD3=optimal_AS=0
```

**Note:** Specify all fields during updates. Fields that are not specified will have no content after the update. Preset fields are disregarded. Key fields are not verified!

## UpdateArchiveDataS

Corresponds to 'UpdateArchiveData' and can be used from VB Script.

**Definition**

UpdateArchiveDataS (String strFilename,  
VARIANT\* varReturnObjectID,  
VARIANT\* varReturnObjectType)

## UpdateDocFileList

<b>Definition</b>	UpdateDocFileList (String strFileName)
<b>Type</b>	Method
<b>Description</b>	Allows the file list of a document to be changed.
<b>Parameters</b>	<b>strFileName</b> file name, for structure see comment
<b>Return value</b>	<p>0 no error</p> <p>-2 no cabinet specified</p> <p>-3 cabinet unknown</p> <p>-6 no document type specified</p> <p>-7 document type unknown</p> <p>-9 document type does not belong to the specified cabinet</p> <p>-11 document identifier missing.</p> <p>-21 document already archived</p> <p>-32 handoff file does not exist</p> <p>-41 file name not specified</p> <p>-64 server error occurred</p> <p>Error text can be determined using <b>GetLastError()</b>.</p>

**Comment** The handoff file has the following format:

```
[DATEILISTE]
SCHRANK=
DOKUMENT=
DOKUMENT-ID=
MAINTYPE=
DATEI1=
DATEI2=
.
.
DATEIn=
ARCHIVIERBAR=[0;1]
```

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

**Note:** The main type of the destination document must be defined with **MAINTYPE=#** where # stands for the main type number. Possible main types are described in the chapter **Introduction**.

**Options** **ARCHIVIERBAR=[0;1]**  
Here the archive status of the document can be changed.

0 not ready to archive  
1 archivable  
1 is the default value

**DATEILÖSCHEN=[0;1]**  
Deletes the given source files.  
0 is the default value

**Example** Source code example:

```
HelpStr=MyAX. UpdateDocFileList(strFile)
UpdateDocFileList(strFile)
```

**Note:** The given file names must not be identical with the file names that have already been assigned to the document. Otherwise data loss will occur.

The file list can only be changed if the document has not been archived!

## UpdateDocShare

<b>Definition</b>	UpdateDocShare (long IIdent, long IType, long IUser)
<b>Type</b>	Method
<b>Description</b>	Edits the release of the specified document for the specified user
<b>Parameters</b>	<b>IIdent</b> Document ID <b>IType</b> Document Type <b>Rights</b> Rights set as defaults (RWXU) <b>IUser</b> User ID of the user of the share
<b>Return value</b>	<b>0</b> "Release edited or editing canceled." <b>-125</b> "A modal dialog is opened." <b>-130</b> "Specified DMS object type is unknown." <b>-131</b> "Specified DMS object type if not a document." <b>-132</b> "Only 'RXWU' rights are permitted." <b>-133</b> "The specified document does not exist or was deleted." <b>-134</b> "Document shares not possible." <b>-135</b> "The system role required to share documents is not present." <b>-136</b> "Specified user does not exist." <b>-139</b> "Specified object was not shared by logged-in user."

Error exact text can be determined using **GetLastError()**.

**Comment** -

### Example

```

dim Application : set Application =
createobject("optimal_AS.application")
Dim sRet

Application.ActivateApp 1

sRet = Application.UpdateDocShare(590, 262146, 15475)

if (sRet <> 0) then
    MsgBox Application.GetLastError() & " (" & sRet & ")"
else
    MsgBox "Share edited or editing canceled."
end if

set Application = nothing
  
```

## UpdateDocumentData

<b>Definition</b>	UpdateDocumentData (String strFilename)
<b>Type</b>	Method
<b>Description</b>	Performs an update of the document data.
<b>Parameters</b>	<b>strFilename</b> handoff file (structure see comment)
<b>Return value</b>	<p><b>0</b> if no error</p> <p><b>-2</b> no folder specified</p> <p><b>-3</b> folder unknown</p> <p><b>-4</b> no register specified</p> <p><b>-5</b> register unknown</p> <p><b>-7</b> unknown document type</p> <p><b>-8</b> register does not belong to the specified folder</p> <p><b>-11</b> document identifier missing.</p> <p><b>-13</b> register identifier unknown</p> <p><b>-16</b> update failed</p> <p><b>-24</b> field names could not be resolved</p> <p><b>-28</b> invalid field value for a given field</p> <p><b>-30</b> one or more mandatory fields not filled out</p> <p><b>-31</b> data type of entered value is not compatible with the data type of the associated field.</p> <p><b>-32</b> handoff file does not exist</p> <p><b>-47</b> no write permission for this object</p> <p><b>-64</b> server error occurred</p> <p>Error text can be determined using <b>GetLastError()</b>.</p>

**Comment** The handoff file has the following structure:

```
[AKTUALISIEREN]
SCHRANK=
DOKUMENT=
DOKUMENT-ID=
FIELD1=
FIELD2=
.
FIELDn=
Mode=1
```

If mode = 1, all non-specified fields are not reset. See also the **note** on UpdateArchiveData

The following lines must be inserted in order to update the table controls, where the separator character equals chr(17).

```
[Table@TABLENAME]
Mode=0[1] 0 = Tabelle leeren, 1 = Tabelle anhängen
Row0={Column1Row1 (Separator) Column2Row1}
Row1={Column1Row2 (Separator) Column2Row2}
```

**Example:**

```
[Tabelle@Protokoll]
Mode=1
Line0={01.02.2013 (Separator) Mustermann}
Line1={04.02.2013 (Separator) Müller}
Line2={05.02.2013 (Separator) Meyer}
```

**The main type of documents which are part of the document history must not be changed.**

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

## Options

### **PFLICHTFELDER=[0;1]**

Mandatory fields are verified for the value 1.

### **REFRESHMULTIFIELDS=[0;1]**

If the value is 1, all entries of the multiple fields, for which new values in the section MULTI\_... were defined, are deleted. If the handoff file contains the value REFRESHMULTIFIELDS=1 and a MULTI\_Field1 section was defined, then all entries are deleted for this field and values entered in this section are inserted.

The database name of the multiple field stands for the update of multiple fields MULTI\_... for ...

```
DATA1=Seitennummer,Wert
DATA2=Seitennummer,Wert
```

### **ARCHIVIERBAR = [0;1]**

here the archive status of the document can be changed.

0 not ready to archive

1 archivable

### **SHOWTEMPLATES=[0;1]**

The template dialog for W\_Documents is displayed if the value is 1 .

The selected template is inserted.

### ***ENABLEOPTIONS=[0;1]***

If the value is 1, the options of the template dialog are activated.

## Example

### Example: Updating a customer document **inbound document**

```
[AKTUALISIEREN]
SCHRANK=Kunde
REGISTER=Register
DOKUMENT-ID=4713
FELD1=Typ=Brief
FELD2=Text=Hallo
[MULTI_FIELD1]
DATA1=1, Peter
DATA2=2, Hans
```

## UpdateRegisterData

<b>Definition</b>	UpdateRegisterData (String strFilename)
<b>Type</b>	Method
<b>Description</b>	Performs an update of the register data.
<b>Parameters</b>	<b>strFilename</b> handoff file (structure see comment)
<b>Return value</b>	<p>0 if no error</p> <p>-2 no folder specified</p> <p>-3 folder unknown</p> <p>-4 no register specified</p> <p>-5 register unknown</p> <p>-8 register does not belong to the specified folder</p> <p>-12 register identifier missing</p> <p>-13 register identifier unknown</p> <p>-16 update failed</p> <p>-24 field names could not be resolved</p> <p>-28 invalid field value for a given field</p> <p>-30 one or more mandatory fields not filled out</p> <p>-31 data type of entered value is not compatible with the data type of the associated field.</p> <p>-32 handoff file does not exist</p> <p>-47 no write permission for this object</p> <p>-64 server error occurred</p> <p>Error text can be determined using <b>GetLastError()</b>.</p>

**Comment** The handoff file has the following structure:

```
[AKTUALISIEREN]
SCHRANK=
REGISTER=
REGISTER-ID=
FIELD1=
FIELD2=
.
.
FIELDn=
Mode=1
```

If mode = 1, all non-specified fields are not reset. See also the note on UpdateArchiveData

The following lines must be inserted in order to update the table controls, where the separator character equals chr(17).

```
[Table@TABLENAME]
Mode=0[1] 0 = Tabelle leeren, 1 = Tabelle anhängen
Row0={Column1Row1 (Separator) Column2Row1}
Row1={Column1Row2 (Separator) Column2Row2}
```

The **strFileName** parameter value can be a file name or instead a string containing the content of the file which would be passed. If file content is passed as a string, each line must end with a line break.

**Options** **PFLICHTFELDER=[0;1]**  
Mandatory fields are verified for the value 1.

**Example**

**Customer register update**

```
[AKTUALISIEREN]  
SCHRANK=Kunde  
REGISTER=Register  
REGISTER-ID=4711  
FELD1=Typ=Brief
```

# COM interface of preview window

## Introduction

Detail and content preview of the client can be addressed and controlled using a COM interface. This interface can either be accessed via the 'application' interface or used directly as long as this happens within an event. Within events this object can be addressed under the name 'InfoWindow'.

## All COM Commands

### Caption

<b>Definition</b>	String Caption
<b>Type</b>	Property (read/write)
<b>Description</b>	Sets the window title or returns it.
<b>Parameters</b>	-
<b>Return value</b>	-

### Example

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.Caption = "This is the caption"
```

## Visible

<b>Definition</b>	Boolean Visible
<b>Type</b>	Property (read/write)
<b>Description</b>	Sets the visibility of the window
<b>Parameters</b>	-
<b>Return value</b>	-

### Example

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.Visible = false 'makes the window invisible
```

## URL

<b>Definition</b>	String URL
<b>Type</b>	Property (read/write)
<b>Description</b>	Sets the URL to be displayed.
<b>Parameters</b>	-
<b>Return value</b>	-

### Example

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.Url = "www.google.de"
```

## Closeable

<b>Definition</b>	Boolean Closeable
<b>Type</b>	Property (read/write)
<b>Description</b>	Specifies if the window can be closed by the user.
<b>Parameters</b>	-
<b>Return value</b>	-

**Example**

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.Closeable = false
```

## HtmlDocument

<b>Definition</b>	Object HtmlDocument
<b>Type</b>	Property (read)
<b>Description</b>	Provides the HTML-DOM document which is currently displayed.
<b>Parameters</b>	-
<b>Return value</b>	-
<b>Comment</b>	The HTML document interface is described on MSDN.

**Example**

```
Dim a as object
Set a = createobject("optimal_as.application")
Set doc = a.InfoWindow.HtmlDocument
```

## EnableContextMenu

<b>Definition</b>	Boolean EnableContextMenu
<b>Type</b>	Property (read/write)
<b>Description</b>	Specifies if the context menu can be displayed
<b>Parameters</b>	-
<b>Return value</b>	-

**Example**

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.EnableContextMenu = false
```

## Refresh

<b>Definition</b>	Refresh()
<b>Type</b>	Method
<b>Description</b>	Updates the view in the window
<b>Parameters</b>	-
<b>Return value</b>	-

### Example

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.Refresh
```

## GoBack

<b>Definition</b>	GoBack()
<b>Type</b>	Method
<b>Description</b>	Navigates to the previous URL.
<b>Parameters</b>	-
<b>Return value</b>	-

### Example

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.URL = "www.google.de"
a.InfoWindow.URL = "www.test.de"
a.InfoWindow.GoBack
```

## GoForward

**Definition**

GoForward()

**Type**

Method

**Description**

Navigates in the history to the next URL.

**Parameters**

-

**Return value**

-

**Example**

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.URL = "www.google.de"
a.InfoWindow.URL = "www.test.de"
a.InfoWindow.GoBack
a.InfoWindow.GoForward
```

## ShowHtml

<b>Definition</b>	ShowHtml(String strHtml)
<b>Type</b>	Method
<b>Description</b>	Displays the passed HTML string.
<b>Parameters</b>	-
<b>Return value</b>	-

### Example

```
Dim a as object
Set a = createobject("optimal_as.application")
a.InfoWindow.ShowHtml
"<html></head><body>Test</body></html>"
```

# Script interface of preview windows

## Introduction

This interface provides you with features that allow you to control the client from the detail and content preview windows, for example in order to refresh a hit list or open a data sheet. All functions can be used in multiple iterations.

## Functions to Browse Across Documents

### osjxCanNextDoc

<b>Definition</b>	Boolean osjxCanNextDoc
<b>Type</b>	Property (read/write)
<b>Description</b>	Indicates whether there is a next document in the current hit list.
<b>Parameters</b>	-
<b>Return value</b>	<b>1</b> There is a next document in the current hit list. <b>0</b> There is no next document in the current hit list.

**osjxCanPrevDoc**

<b>Definition</b>	Boolean osjxCanPrevDoc				
<b>Type</b>	Property (read/write)				
<b>Description</b>	Indicates whether there is a previous document in the current hit list.				
<b>Parameters</b>	-				
<b>Return value</b>	<table><tr><td><b>1</b></td><td>There is a previous document in the current hit list.</td></tr><tr><td><b>0</b></td><td>There is no previous document in the current hit list.</td></tr></table>	<b>1</b>	There is a previous document in the current hit list.	<b>0</b>	There is no previous document in the current hit list.
<b>1</b>	There is a previous document in the current hit list.				
<b>0</b>	There is no previous document in the current hit list.				

## osjxNextDoc

<b>Definition</b>	osjxNextDoc()
<b>Type</b>	Method
<b>Description</b>	Sets the focus on the next document.
<b>Parameters</b>	-
<b>Return value</b>	-

## osjxPrevDoc

<b>Definition</b>	osjxPrevDoc()
<b>Type</b>	Method
<b>Description</b>	Sets the focus on the previous document.
<b>Parameters</b>	-
<b>Return value</b>	-

## Functions to Control Client Windows

### osjxOpenDataSheet

<b>Definition</b>	osjxOpenDataSheet (long lObjectID, BOOL bReadOnly )
<b>Type</b>	Method
<b>Description</b>	Opens the data sheet of a specified object.
<b>Parameters</b>	<b>lObjectID</b> object ID <b>bReadOnly</b> TRUE if the document is to be opened read-only
<b>Return value</b>	-
<b>Comment</b>	-

### osjxOpenObject

<b>Definition</b>	osjxOpenObject(long lObjectID)
<b>Type</b>	Method
<b>Description</b>	Opens the document or the folder of an object.
<b>Parameters</b>	<b>lObjectID</b> object ID
<b>Return value</b>	-

**Comment** -

## osjxOpenLocation

**Definition** osjxOpenLocation(long lObjectID)  
**Type** Method  
**Description** Opens the location of a selected object.  
**Parameters** **lObjectID** object ID  
**Return value** -  
**Comment** -

## osjxOpenLocationsAndLinks

**Definition** osjxOpenLocationsAndLinks(long lObjectID)  
**Type** Method  
**Description** Opens links and references of an object.  
**Parameters** **lObjectID** object ID  
 — **Return value** -

## osjxOpenObjectHistory

**Definition** osjxOpenObjectHistory(long lObjectID)  
**Type** Method  
**Description** Opens the editing history of an object.  
**Parameters** **lObjectID** object ID  
**Return value** -

## osjxAddSignature

**Definition** osjxAddSignature(long lObjectID)  
**Type** Method  
**Description** Adds an electronic signature.  
**Parameters** **lObjectID** object ID  
**Return value** -

## osjxPrintObject

<b>Definition</b>	osjxPrintObject()
<b>Type</b>	Method
<b>Description</b>	Prints a document.
<b>Parameters</b>	-
<b>Return value</b>	-

## osjxOpenObjectRemarks

<b>Definition</b>	osjxOpenObjectRemarks(long lObjectID)
<b>Type</b>	Method
<b>Description</b>	Opens the notes of an object.
<b>Parameters</b>	<b>lObjectID</b> object ID
<b>Return value</b>	-

## osjxAddFollowUp

<b>Definition</b>	osjxAddFollowUp(long lObjectID)
<b>Type</b>	Method
<b>Description</b>	Adds a follow-up.
<b>Parameters</b>	<b>lObjectID</b> object ID
<b>Return value</b>	-

## osjxAddSubscribe

<b>Definition</b>	osjxAddSubscribe(long lObjectID)
<b>Type</b>	Method
<b>Description</b>	Adds a subscription.
<b>Parameters</b>	<b>lObjectID</b> object ID
<b>Return value</b>	-

## osjxStartWorkflow

<b>Definition</b>	osjxStartWorkflow    (long String    lObjectID, strWorkflowModell)
-------------------	--

<b>Type</b>	Method
<b>Description</b>	Starts a workflow with a given object, based on the model name.
<b>Parameters</b>	<b>IObjectID</b> object ID <b>strWorkflowModell</b> Name of the workflow model
<b>Return value</b>	-

## osjxGetSelectedObjects

<b>Definition</b>	osjxGetSelectedObjects()
<b>Type</b>	Method
<b>Description</b>	Finds desired objects in a hit list.
<b>Parameters</b>	-
<b>Return value</b>	List of selected objects: Obj0-ID, Obj0-Type; ...; Obj (n) -ID, Obj (n) -Type

## osjxRefreshObjectInLists

<b>Definition</b>	osjxRefreshObjectInLists (long IObjectID)
<b>Type</b>	Method
<b>Description</b>	Updates the specified DMS object in all open lists.
<b>Parameters</b>	<b>IObjectID</b> object ID
<b>Return value</b>	-
<b>Comment</b>	-
<b>Example</b>	<pre>if (window.osClient) { window.osClient.osjxRefreshObjectInLists(312); } else { alert("window.osClient not exist"); }</pre>

## osjxByteArrayToFile

<b>Definition</b>	osjxByteArrayToFile(JavascriptArray, sting sDateiname)
<b>Type</b>	Method
<b>Description</b>	Receives a JavaScript array with integer values and writes them to a file.

<b>Parameters</b>	<b>JavaScript array</b> <b>sDateiname</b>
<b>Return value</b>	-
<b>Comment</b>	If the file name is specified without a unique path, a 'Save under' dialog will be opened.

<b>Example</b>	<pre> if (window.osClient) { var arr = new array(10);  arr[0] = 0; arr[0] = 1; arr[0] = 2; arr[0] = 3; arr[0] = 4;  window.osClient.osjxByteArrayToFile(arr, "file.pdf"); } else { alert("window.osClient not exist"); } </pre>
----------------	---

## osjxURLDownloadToFile

<b>Definition</b>	osjxURLDownloadToFile(string sDateiUrl, sting sDateiname)
<b>Type</b>	Method
<b>Description</b>	Receives a file ULR and writes the content of the file ULR to a file.
<b>Parameters</b>	<b>sDateiUrl</b> <b>sDateiname</b>
<b>Return value</b>	-
<b>Comment</b>	If the file name is specified without a unique path, a 'Save under' dialog will be opened.

<b>Example</b>	<pre> if (window.osClient) { window.osClient.osjxURLDownloadToFile ("http://www.url.de/bild.jpg", "urlbild.jpg"); } else { alert("window.osClient not exist"); } </pre>
----------------	---

## osjxOpenResultList

<b>Definition</b>	osjxOpenResultList(sting sTitel, JSON-String)
<b>Type</b>	Method

**Description** Opens a hit list with the specified title and the specified DMS objects.

**Parameters** **sTitel**  
**JSON-String**

**Return value** -

**Example** JSON-String:

```
{
  "title": "my folder hit list",
  "hits": [{
    "id": "411",
    "type": "8"
  },
  {
    "id": "577",
    "type": "8"
  }
]}
```

**Call:**

```
if (window.osClient)
{
window.osClient.osjxOpenResultList("{\"title\": \"Trefferliste\", \"hits\": [{\"id\": \"411\", \"type\": \"8\"}, {\"id\": \"577\", \"type\": \"8\"}]}");
}
else
{
alert("window.osClient not exist");
}
```

## Functions for Dashlet Control

Dashlets can be identified in three ways:

1. No parameter: the active dashlet
2. Dashlet title: The title specified in enaio® enterprise-manager when setting up the dashlet. The title 'OSDOCUMENTVIEWER' or 'OSDETAILVIEWER' is used to access the system-side dashlets 'DocumentViewer' or 'DetailsViewer'.
3. Number of the dashlet (1 – 10)

In the following, this function parameter is marked as 'dashletID'.

### osjxOpenChromeDEVTools

**Definition** osjxOpenChromeDEVTools()

**Type** Method

**Description** Opens DEV tools of the Chrome browser.

**Parameters** -

**Return value** -

## osjxCloseChromeDEVTools

<b>Definition</b>	osjxCloseChromeDEVTools()
<b>Type</b>	Method
<b>Description</b>	If the DEV tools of the Chrome browser are opened, they can be closed using this method.
<b>Parameters</b>	-
<b>Return value</b>	-

## osjxIsDashletVisible

<b>Definition</b>	osjxIsDashletVisible(dashletID)
<b>Type</b>	Method
<b>Description</b>	Determines if the specified dashlet is visible.
<b>Parameters</b>	<b>dashletID</b> (see above) Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used.
<b>Return value</b>	<b>1</b> Dashlet is visible <b>0</b> Dashlet is not visible

## osjxSetDashletVisible

<b>Definition</b>	osjxSetDashletVisible(dashletID, bool bVisible)
<b>Type</b>	Method
<b>Description</b>	Makes the specified dashlet visible or not visible. If it is not visible, it cannot be reached from enaio® client via the 'View' ribbon.
<b>Parameters</b>	<b>dashletID</b> (see above) Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used. <b>bVisible</b> 0 = not visible / 1 = visible
<b>Return value</b>	-

## osjxIsDashletEnabled

<b>Definition</b>	osjxIsDashletEnabled(dashletID)
<b>Type</b>	Method
<b>Description</b>	Determines if the specified dashlet is active.
<b>Parameters</b>	<b>dashletID</b> (see above)

Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used.

**Return value**

<b>1</b>	Dashlet is active
<b>0</b>	Dashlet is not active

## osjxSetDashletEnabled

**Definition** osjxSetDashletEnabled(dashletID, bool bEnabled)

**Type** Method

**Description** Sets the specified dashlet to active or inactive  
If inactive, 'ContextChanges' are ignored.

**Parameters** **dashletID** (see above)  
Dashlet name as string or number of the dashlet. If none is specified, the current dashlet is used.  
**bEnabled** 0 = inactive / 1 = active

**Return value** -

## osjxGetDashletURL

**Definition** osjxGetDashletURL(dashletID)

**Type** Method

**Description** Determines the URL of the specified dashlet.

**Parameters** **dashletID** (see above)  
Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used.

**Return value** URL of the dashlet as string

## osjxSetDashletURL

**Definition** osjxSetDashletURL(dashletID, string sUrl)

**Type** Method

**Description** Sets the URL of the specified dashlet. If no URL is specified, the originally configured URL is set.

**Parameters** **dashletID** (see above)  
Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used.  
**sUrl** URL of the dashlet as string

**Return value** -

## osjxDashletPaneState

<b>Definition</b>	osjxDashletPaneState(dashletID)
<b>Type</b>	Method
<b>Description</b>	Determines the display status of the area of the specified dashlet.
<b>Parameters</b>	<b>dashletID</b> (see above) Dashlet name as string or number of the dashlet. If none is specified, the current dashlet is used.
<b>Return value</b>	0: unknown 1: closed 2: hidden 3: floating

## osjxSetDashletPaneState

<b>Definition</b>	osjxSetDashletPaneState(dashletID, long lStatus)
<b>Type</b>	Method
<b>Description</b>	Sets the display status of the area of the specified dashlet. If the specified dashlet is set to not visible using 'osjxSetDashletVisible', invoking the method is ignored.
<b>Parameters</b>	<b>dashletID</b> (see above) Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used. <b>lStatus:</b> 0: unknown 1: closed 2: hidden 3: floating
<b>Return value</b>	-

## osjxSetDashletCaption

<b>Definition</b>	osjxSetDashletCaption(dashletID, string sTitle)
<b>Type</b>	Method
<b>Description</b>	Specifies a title for a dashlet.
<b>Parameters</b>	<b>dashletID</b> (see above) Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used. <b>sTitel</b>
<b>Return value</b>	-
<b>Example</b>	<pre>if (window.osClient) {</pre>

```
window.osClient.osjxSetDashletCaption
("OSDETAILVIEWER", "New Title");
}
else
{
alert("window.osClient not exist");
}
```

## osjxGetDashletCaption

<b>Definition</b>	osjxGetDashletCaption(dashletID)
<b>Type</b>	Method
<b>Description</b>	Determines the title of a dashlet.
<b>Parameters</b>	<b>dashletID</b> (see above) Dashlet title as string or number of the dashlet. If none is specified, the current dashlet is used.
<b>Return value</b>	Title of the dashlet.
<b>Example</b>	<pre>if (window.osClient) { var State = window.osClient.osjxGetDashletCaption("OSDETAILVIEWER" );  alert(State); } else { alert("window.osClient not exist"); }</pre>

## osjxGetEnvironment

Determines values known from the COM features 'GetEnvironment'.

# Appendix: Configuring the enaio® Printers

## Introduction

When installing enaio® client, optionally two printer drivers can be installed at the workstation:

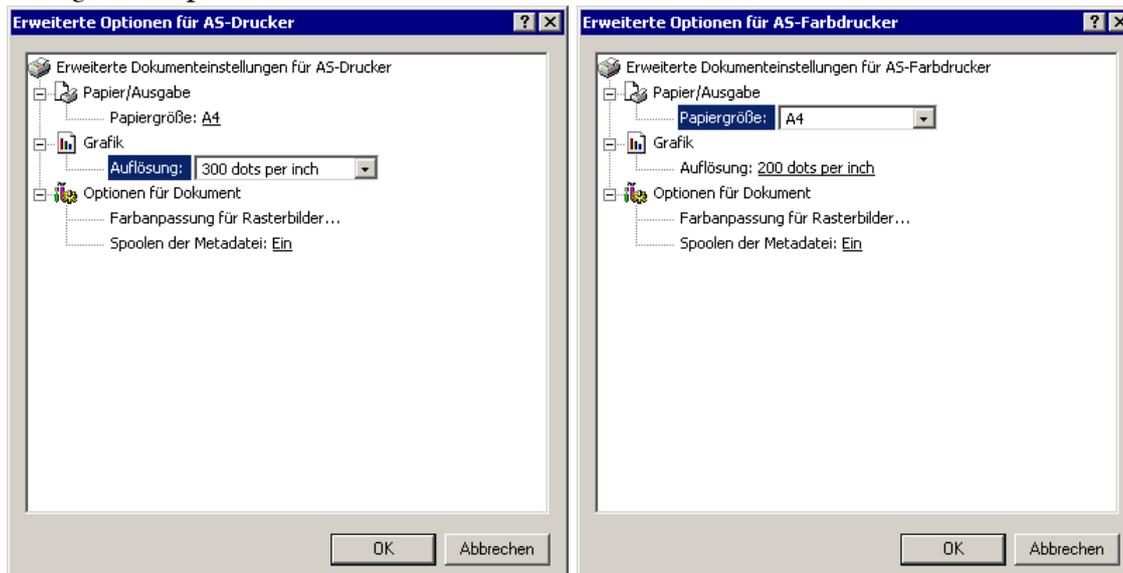
- OS-printer `asprint.dll` for black/white printing
- the OS color printer `axcprint.dll` for color printing

If you have not installed these printer drivers, you can obtain a printer driver setup upon request. Using these printer drivers, image files can be created from any application with printing abilities and passed on to enaio® client. enaio® client opens the corresponding dialogs in which the user can specify a location, document type, and indexing. It is required to have enaio® client launched. If it has not been launched an information dialog will open. The user can then launch enaio® client and continue the process, or cancel printing.

The printer drivers can also be used to save printing data as image or PDF-files in the file system. The configuration file `asprint.ini` will be required.

This configuration file must be created by the corresponding application and saved in the Windows directory. Usually, after printing the configuration file must be deleted.

Paper size and resolution are not configured in the configuration file but as usual in the properties dialogs of the printer drivers.



**Note:** The OS color printer must not be set to grayscale printing. This setting will cause errors.

## The Configuration File

The printer drivers locate the configuration file `asprint.ini` in the Windows directory before creating any files.

Only if the file does not exist, is an attempt made to pass on image files to enaio® client. If the file is found, the configuration data will be read.

You can create the file with any text editor. It must be labeled `asprint.ini` and saved in the Windows directory.

Enter the section label `[ASPRINT]` in the first row of the file. Followed by the parameters:

Parameters	Possible values	Description
FILE=	<Path\FileName>.000	Enter path, file name and extension. The printer drivers always use an automatic hexadecimal extension if several pages are created as single files. The first page has the extension '000'.
	<Path\FileName>.tif	If you specify the extension 'tif', the AS-printer will use this extension if all pages are saved as one multipage TIFF G4 file. Using MULTIPAGE=1 creates one file for each print job. Otherwise automatically the extension 000
	<Path\FileName>.pdf	If you specify the extension 'pdf', the OS-printer and OS color printer will use this extension if all pages are saved as one PDF file.
HINTERGRUND1=	<Path\FileName>.tif	The OS-printer can include background images. The files must be in TIFF G4 format and it is recommended that they have the same resolution as the one that is defined for the OS printer. This parameter defines the background image for the first page.
HINTERGRUND2=	<Filename>.tif	This parameter defines the background image for all following pages.
MULTIFILE=	0	If there are already files with the same label, they are not overwritten. The print job will be canceled. The result code '-8 – file already exists' is inserted into the log file.
	1	Depending on the parameter 'Multipage' files with the same label are not overwritten, but their extension is incremented or the file is extended.
MULTIPAGE=	0	A file is created for each page. The extension starts as hexadecimal '000' and is incremented. Using the parameter 'MULTIFILE=1' the printer drivers determine if files of the same name already exist and in such case continue with the numbering of the extensions. With parameter 'MULTIFILE=0' the print job is canceled if files with the same extension already exist.
	1	The OS-printer creates a multipage TIFF G4 file. If this file already exists, new pages are added. If the parameter has the value '1', the file is created with a PDF header and can be opened in a PDF viewer. The OS color printer creates a PDF-file with all pages if the 'PDF' parameter has the value '1'. If this file already exists, new pages are added. If the 'PDF' parameter has the value '0', a JPEG file is created for each page. The extension starts as hexadecimal '000' and is incremented.

Parameters	Possible values	Description
PDF=	0	The OS-printer creates image files in TIFF G4 format. The OS color printer creates color images in JPEG-format. The default setting is '0'.
	1	The created image files get a PDF-header and can be opened in a PDF viewer.
GRAY=	0	Default setting of the OS color printer, color images are created.
	1	The OS color printer creates grayscale images in JPEG-format.
COMPRESSION=	1-100	The level of compression can be adjusted for the OS color printer. A value of '100' denotes maximum compression. The default setting is '1' – minimal compression.
EXE=	<Path\Program>	Path and label of a program that will be started after the print job. On launch, the parameters are passed to the program in quotation marks. <ul style="list-style-type: none"> <li>▪ Path and label of the configuration file asprint.ini</li> <li>▪ Log entries (see Logging)</li> </ul> In addition the user name is entered into the asprint.ini file: USERNAME='Windows user name'
CITRIXMODE=	1	This entry is only required for a terminal server installation (see Terminal Server)

**Example:**

```
[ASPRINT]
FILE=C:\ASDRUCK\print.tif
BACKGROUND1=C:\ASDDRUCK\HG1.tif
HINTERGRUND2C:\ASDDRUCK\HGff.tif
MULTIFILE=1
MULTIPAGE=1
PDF=0
```

The OS-printer creates the file `print.tif`. The format is multipage TIFF G4. The file is extended in case of the following print jobs.

## Logging

The enaio® printer drivers create a LOG file with a result number and a description. The LOG-file has the same label as the image file and the file extension 'LOG'. It can be opened with any editor.

Number	Description
1	"Pages were created successfully"
-1	"Erroneous or invalid parameter."
-2	"Not enough memory available."
-3	"Error while locking memory area."
-4	"Error creating a file."
-5	"File does not exist or could not be opened."
-6	"Error reading a file."
-7	"Error writing a file."

-8	"File already exists."
-9	"Incorrect or unsupported file format."
-10	"Incorrect or unsupported TIFF format."
-23	"Error creating a directory."
-24	"Invalid destination directory."
-25	"Invalid source directory."
-26	"Invalid temporary directory."
-27	"Invalid drive."
-28	"Insufficient hard disk space."
-29	"Error switching the directory."
-30	"Error while deleting a directory."
-31	"Error determining the file size in a directory."
-44	"Error while loading a DIB."
-45	"DIB is faulty or does not exist."
-46	"Error while creating a bitmap."
-73	"No file name available for background bitmap"
-74	"File does not contain image data."
-78	"Unknown error."
-79	"File does not exist."
-80	"Source file could not be opened."
-81	"Destination file could not be opened."
-82	"Source file does not exist."
-83	"Destination file not found."
-85	"Error creating a slide."
-91	"Action canceled by user."

**Example:**

```
-8: file C:\ASDRUCK\print.000 already exists
```

If an application is run after printing, this content, the path, and the name will be passed to the configuration file `asprint.ini` as a start parameter.

## Terminal Server

When installing terminal servers both printer drivers require the configuration file `asprint.ini`. The path to the configuration file is entered in the registry under:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Optimal Systems
```

There create the key 'asprinter'. Enter 'directory' for string and as value a directory that will be accessible for the user.

Places the configuration file `asprint.ini` in this directory. The parameter 'CITRIXMODE=1' will be required.

Enter any file label for the 'file' parameter. Path and file name are not process, but possibly the file extension.

The printer drivers always save files according to the following schema.

- Inside the directory containing the configuration file for each user that prints a subdirectory is created in which the files are stored.
- The sub-directory is labeled with the Windows user name.
- The files are labeled with the Windows user name. The extension depends on the parameters.

The other parameters have the same functions in case of terminal server installations. Also a LOG file is created.

# Appendix: Structure Tree Catalogs

## File Conversion

The data of structure tree catalogs is administered with structure tree files. These files are created with the enaio® editor.

Since version 5.5 an API interface is available with the library `oxlist.dll` in order to convert ASCII text files into structure tree files and structure tree files into ASCII text files.

This allows data for the structure tree catalog to be created dynamically, to be updated and synchronized.

The library `oxlist.dll` is copied along with the also required library `oxmisc.dll` into the installation directories `clients\client32` and `clients\admin`.

The interface contains the two following functions which can e.g. be addressed using Visual Basic:

```
int WINAPI ConvertAsciiFileToTreeFile(LPCSTR lpAsciifile, LPCSTR lpDefinition, LPCSTR
lpTreefile);
/*!
  The function generates a structure tree file from an ASCII-file and a corresponding
  layer definition.

  @param LPCSTR lpAsciifile    - ASCII file that contains sorted short forms and
  corresponding labels. (short form and labels must be separated by spaces, not by tabs)
  @param LPCSTR lpDefinition   - level definition, e.g. AA@A.9.AA
  (A -> letters,numbers 9 -> numbers only, @ -> no separator, . -> separator)

  @param LPCSTR lpTreefile     - file name of the created structure tree file.

  @return int                  - 0 -> if successful, else -> error code
*/

int WINAPI ConvertTreeFileToAsciiFile(LPCSTR lpTreefile, LPCSTR lpAsciifile);
/*!
  The function generates a ASCII-file from a structure tree file.

  @param LPCSTR lpTreefile     - file name of an existing structure tree file.
  @param LPCSTR lpAsciifile    - ASCII-file to be created that contains sorted short
  forms and corresponding labels.

  @return int                  - 0 -> if successful, else -> error code
*/
```

### Example on how to use this functions in Visual Basic:

```
Private Declare Function ConvertTreeFileToAsciiFile Lib "oxlist.dll"
(ByVal lpTreefile As String, ByVal lpAsciifile As String) As Long
Private Declare Function ConvertAsciiFileToTreeFile Lib "oxlist.dll"
(ByVal lpAsciifile As String, ByVal lpDefinition As String,
ByVal lpTreefile As String) As Long

Private Sub Command1_Click()
lRet = ConvertTreeFileToAsciiFile
("C:\\Import\\Struktur.dat", "C:\\Import\\Struktur.txt")
End Sub

Private Sub Command2_Click()
lRet = ConvertAsciiFileToTreeFile
("C:\\Import\\Struktur.txt", "AA-99-AA", "C:\\Import\\Struktur1.dat")
End Sub
```

## Structure of the text file

Specify a layer definition when creating a structure tree file in enaio® editor. When converting an ASCII text file into a structure tree file also specify this layer definition.

Within the ASCII file align short form and entry in each line.

The short form must match the layer definition in terms of number of places and separator position.

Any separator character can be used in the syntax of the ASCII file, only the position is important.

In the example below instead of the separators '/' and '-' simply spaces were inserted.

Do not use a separator in between layers, like the '@' in the layer definition, also do not insert a separator in between layer short forms.

Short form and entry are separated by a space.

Add spaces for all short forms except the short forms of the last layer until the number of spaces matches the layer definition.

### Example:

```
Layer definition: 99/99-A
```

```
File content:
```

```
01      2001
01 01   January
01 01 F Family law
01 01 U Copyright law
01 01 V Contract law
01 02   February
01 02 F Family law
01 02 U Copyright law
01 02 V Contract law
01 03   March
01 03 F Family law
01 03 U Copyright law
01 03 V Contract law
02      2002
02 01   January
```

HIER LOCHEN ODER DIGITAL ARCHIVIEREN

### Illustration:

The number of positions according to the layer definition is seven, each short form has seven positions. Short forms of the first and second layer are complemented with spacing characters. The eighth position is a space and separates short form and entry. A tabulator is not allowed.

The third position is a separator, any separator can be inserted in the ASCII text file. The same applies to the sixth position.

Assignments must be in hierarchical order. An assignment for the second level must follow the corresponding assignment of the first level in the file.