OPTIMAL SYSTEMS

enaio®

Software Documentation
enaio® System Handbook DMS

Version 8.10

18.03.2015
Version 8.10

Content

Foreword

This system handbook is aimed at all those who require information on process and data flows, interdependencies, structures and technologies of the document management system (DMS) enaio® beyond basic operation. It is a useful guide for users and administrators who set up, configure and administer enaio® products.

We have tried to gather and systemize all information which is required to understand the system architecture and interdependencies between the system components. Consequently you should be able to get a better understanding of certain processes and to become aware of the idea, the advantages and disadvantages of specific settings, configurations and installations.

We have tried to avoid repeating information already contained in the user manuals as far as this was possible with regards to the understanding. In some cases you will find references to further descriptions which contain information on parts of the DMS, e.g. database description, application server description, installation description etc.

The system handbook supports the user regarding the configuration, operation and maintenance. This system handbook will also assist you in resolving problems and recognizing errors.

The handbook is divided into two parts. The "Document Management System" is contained in part 1, the "Workflow Management System" in part 2.

General Description of the DMS

A filing tray management system (DMS) is used to capture (import), save (file, archive), manage (search), edit and distribute (export) digital documents. enaio® is a DMS based on the Windows operating system family. In the following the essential topics and their implementation in enaio® will be discussed.

Configurability and Depiction of Information Structures

One of the main advantages of enaio® is its ability to adequately depict information and organizational structures of companies in which it is used, which allows it to adapt to the company's structures and processes and not the other way around.

An unlimited number of possibilities to create information structures by creating cabinets, folders and registers can be achieved with a number of tools provided by enaio®, e.g. enaio® editor, enaio® administrator and enaio® client.

enaio® supports all essential, common document types, e.g.

- pixel graphics (e.g. TIFF, JPEG, BMP),
- e-mails,
- XML and
- all documents created by Windows applications.

Besides the configuration of the information structure, enaio® supports the depiction of the organizational structure of a company and parts of it. In enaio® administrator any number of users and user groups can be set up which match the organizational units of the company.

Document Import, Conversion and Processing

A DMS is used to manage, but not mainly create digital documents. enaio® allows you to manually insert documents in enaio® client and disposes of powerful features to perform automatic import of large numbers of documents. Furthermore, enaio® provides a high number of filter and conversion programs which can be used to process and import documents in different formats (e.g. COLD, DICOM etc.). During such process index data can be captured automatically. The batch capture of invoices in enaio® capture allows for the automated capture of large numbers of invoices. Also, enaio® provides different interfaces for data transfer from other leading systems (SAP, b2, and many more).

Document Processing and Search

enaio® client has its own user interface which allows documents to be edited visually, e.g. adding or deleting pages or changing the order of pages in a document. Also, notes, comments, annotations etc. can be added to the document.

Windows documents are opened in the respective application. For MS Office applications (Word, Excel and PowerPoint) special utilities and macros are provided, which allow you to transfer data from the DMS into a document. With applications, e.g. form printing, automated processing of a large number of similar documents can be performed in e.g. the public administration or health care sector.

Besides the search in the database, enaio® also offers the option of executing a full text search for all information and documents available in the DMS. The OCR of FineReader is supported in this context, allowing you to have all scanned documents queried.

Document Output and Export

Once documents were imported and edited, they must be output, too. enaio® can print and save documents manually in enaio® client. Furthermore, an export module is provided which allows for the automatic export of documents.

Archiving

enaio® can archive documents in a structured way to meet the requirements of permanent archiving. Regular archiving prevents data loss and increases the efficiency by working with up-to-date documents. enaio® offers automated archiving which can be configured to archive documents which fulfill specific criteria regularly and at specific times (e.g. at nighttime) without interrupting the working process.

In this context, enaio® allows for integration with various third-party systems, e.g. iTernity, Centera and iXOS.

Three-Tier Architecture, Availability and Number of Users

Every user wants to access his DMS in any way and large network units must fulfill high demands regarding availability. To meet the requirements of extensive network usage, enaio® was implemented in classic three-tier architecture:

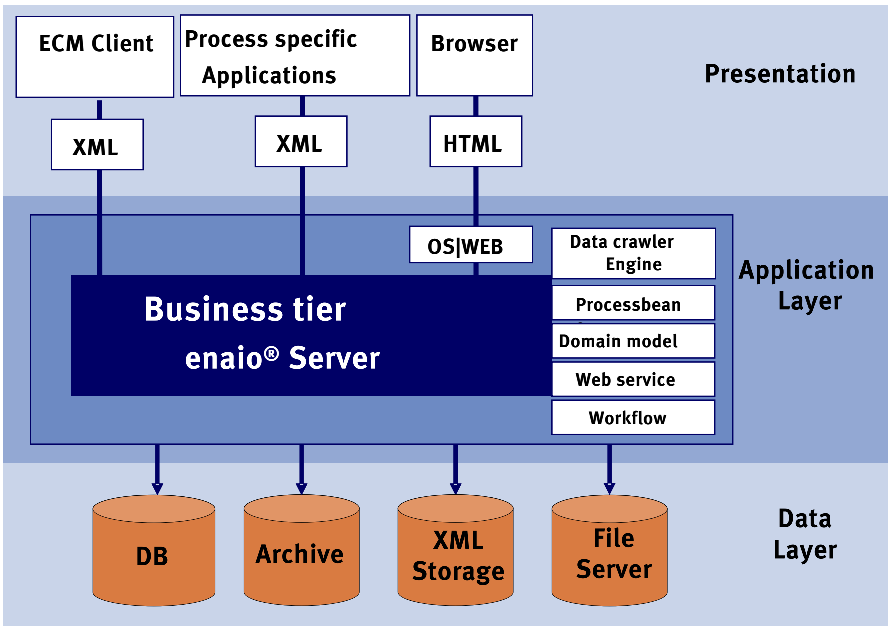


Figure 1: three-tier architecture of enaio

This architecture offers many advantages, in particular for the use in networks. The database can be installed on a designated computer with any supported operating system. An archiving jukebox can also be integrated with any (Windows) computer. The documents do not need to be filed on the computer on which enaio® server (application server) is installed.

Any number of application servers can be installed in a network. These are then combined in server groups with users assigned. Documents and information of the DMS are accessible for all server groups. As a result, enaio® can be scaled almost without limits. In case the already installed application servers reach their performance limit (in most cases the performance limit of the hardware of the application server), simply another server can be added. This makes it possible to react on the growth of users and achieve high availability. The servers and server groups are administered in enaio® enterprise-manager.

Data Security

Data security refers to the prevention of data loss and protection from unauthorized data access. Various tools used to check the consistency of database entries and filed documents which help to identify if a document was lost exist and can be executed automatically and regularly. To prevent unauthorized data access, a complex system for the assignment of access permissions was implemented:

- Only users with a user name and a password set up in the system (login accounts) can access data. Individual accounts can also be locked if login fails three times.
- Users are combined in groups which have different roles assigned for working in the system. Different rights can then be derived for the administration of the system and the execution of various actions.
- Logical expressions can be used to define who can access which documents.

Future-proof

The use of technology standards is of vital importance for the future operation of DMS. Even if proprietary formats and protocols lead to improved performance for a short while, the use of common standards ensures that the system will be in use in the future. Hence, enaio® uses the standard network protocol TCP/IP for its connection between servers and clients. XML RPCs are used which makes the interchange of data and parameters possible. The general XML format makes the server accessible to any platform and over the internet.

System Restrictions

The maximum size of document files which can be handled by enaio® is currently 2 GB. The maximum page count for image documents in PDF format is 50,000 pages. The guaranteed maximum page count for single page tiff is 4096. The possibility of processing such large files creates high requirements for the RAM available on the computers used to display and create these documents. Larger document files cannot be managed with this system.

Further information on the system requirements can be found in the accompanying documents. These contain information on the edge conditions of a supported infrastructure.

enaio® Versions and Configurations

As a rule, configurations are upwards compatible with different enaio® versions, that is, configurations created in earlier system versions can also be used in later system versions.

Downward compatibility is by contrast not guaranteed since enaio® platform developments result in new features and extended configurations. As a consequence, configurations created in later system versions cannot be used in earlier system versions.

It is very possible, that import attempts of configurations which have been created in earlier system versions will result in feature settings getting lost, format incompatibility and/or program errors. Therefore you must pay attention to system versions when dealing with test, development and productive systems and in any case abstain from attempting to import and activate configurations which have been created in later system versions. This applies to configuration files, in particular to object definitions and workflow models.

System Architecture

Database and Document Filing

Database and document filing constitute the data layer of the three-tier architecture.

enaio® uses a database

- for the management of documents,
- for the management of information associated with the documents,
- for the access control of documents and other system resources,
- for the storage and administration of users, user groups, and access permissions.

The operating system or the database manufacturer supplies ODBC drivers that enable communication with the databases.

In enaio® documents are not filed in a database but in a separate file system. This conserves the resources and maintains the performance of the database.

Object Definition, Indexing Form and Object Tables

The object definition in enaio® describes a list of index forms which are created for each of the different DMS objects. These index forms (Figure 2) are created using enaio® editor. These can contain a number of dialog elements which are discussed in the enaio® editor handbook.

enaio® stores the object definition entirely in the database. The following database tables are used for this purpose

- `osobjdef`, object description,
- `osobjfields`, description of dialog elements on the index forms,
- `osobjparrel`, determination of the object hierarchy, i.e. which object can be contained in which (other) DMS object,
- `osconf`, storage of data from `aslisten.dat`,
- `oslistctrl`, description of the database element,
- `osproperties`, other properties of objects and dialog elements.
- `ospagectrl`, description of page controls.

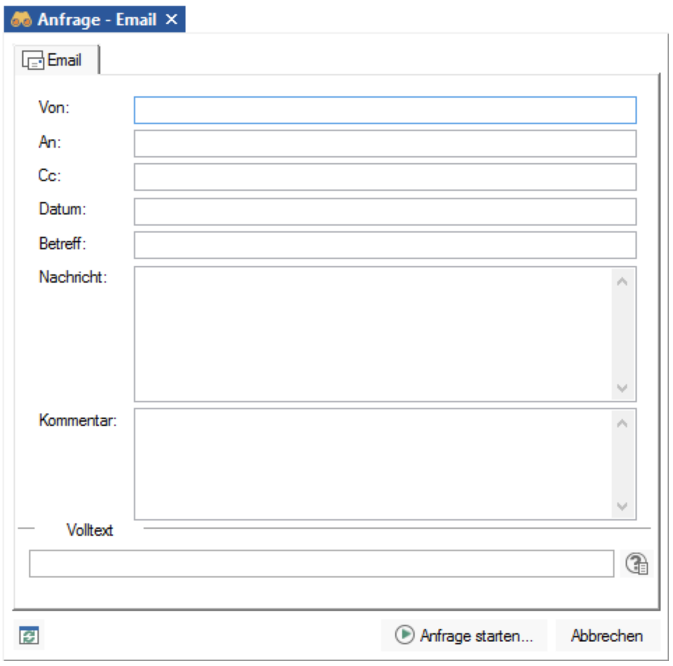


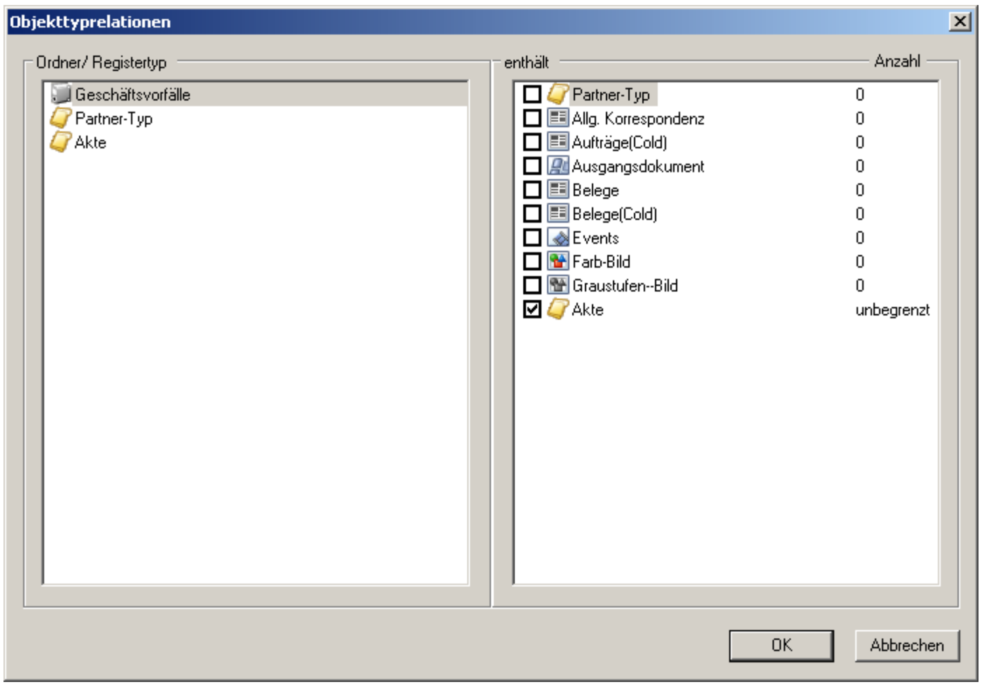
Figure 2: Example of an index form (in this case for e-mails)

enaio® uses the following DMS objects to depict information structures:

- Cabinet – The term 'cabinet' was chosen as a counterpart to a file cabinet which contains folders, registers, and documents. It is the top level in the DMS. All other DMS objects can only exist as sub-objects of a cabinet. A cabinet's only attribute is its name. A database table is not created for a cabinet.
- Folder – For each cabinet one folder type with an index form can be created. Any number of folders can be contained in a cabinet. (Example: cabinet 'Kunden' with folders 'Kunde X')
- Register – Registers are used to create a more detailed information structure. Any number of register types can be created for the object definition.
- Documents – Document types which can be filed in a cabinet are defined using index forms. A document (e.g. a letter, a fax or a scanned newspaper article) can be imported system only by completing (creating) an index form in the enaio® system.

A table is created for each DMS object (see folders, registers and documents below). The object definition can be used to define if and which relations exist between DMS objects of a cabinet. Figure *3* to Figure *5* show how relations between object types of a cabinet may be defined by having the folder 'Geschäftsvorfälle', the register types 'Akte' and 'Partner-Typ' and various document types. Figure 3 shows that only registers with the type 'Akte' can be created in the folder, figure 4 shows that a register with the type 'Akte' can only contain registers with the type 'Partner-Typ', and figure 5 shows that documents can only be filed in registers with the type 'Partner-Typ'. The following object hierarchy is created with these settings:

```
folder   'Geschäftsvorfälle'/register   'Akte'/register   'Partner-Typ'/
document D,P,X,W,XML
```



Objekttyprelationen

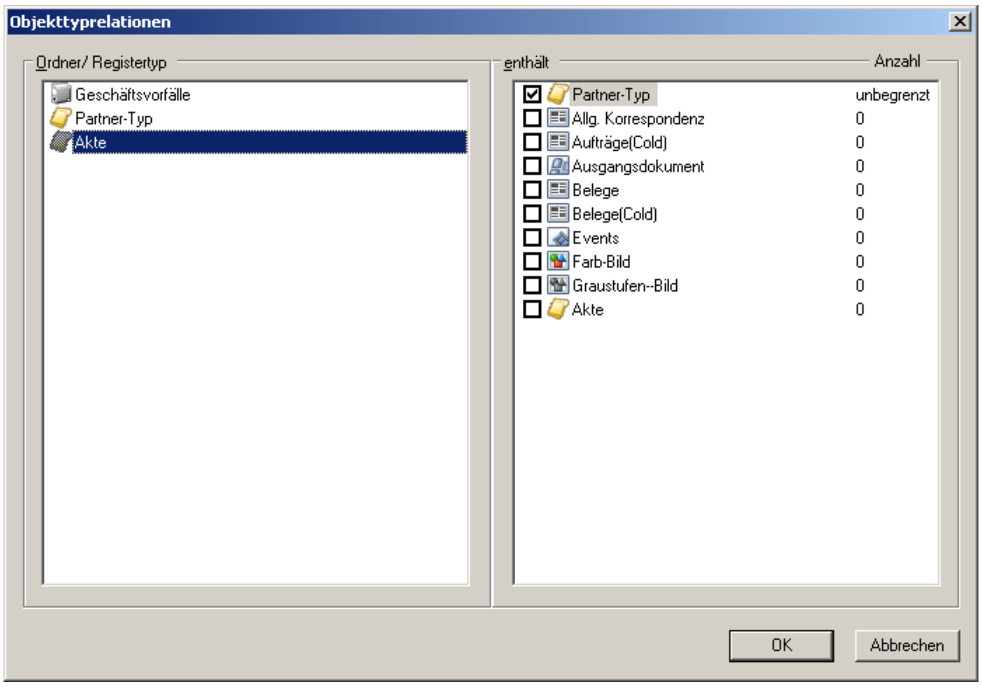
Ordner/ Registertyp

- Geschäftsvorfälle
- Partner-Typ
- Akte

enthält	Anzahl
☐ Partner-Typ	0
☐ Allg. Korrespondenz	0
☐ Aufträge(Cold)	0
☐ Ausgangsdokument	0
☐ Belege	0
☐ Belege(Cold)	0
☐ Events	0
☐ Farb-Bild	0
☐ Graustufen-Bild	0
☑ Akte	unbegrenzt

OK | Abbrechen

Figure 3: Object relation folder – other objects



Objekttyprelationen

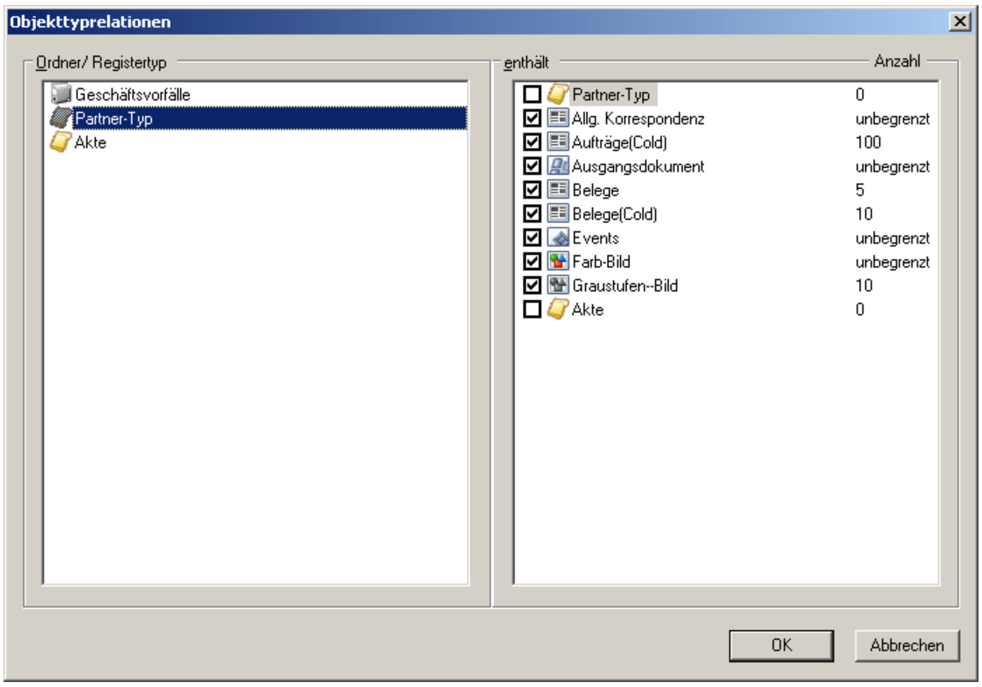
Ordner/ Registertyp

- Geschäftsvorfälle
- Partner-Typ
- Akte

enthält	Anzahl
☑ Partner-Typ	unbegrenzt
☐ Allg. Korrespondenz	0
☐ Aufträge(Cold)	0
☐ Ausgangsdokument	0
☐ Belege	0
☐ Belege(Cold)	0
☐ Events	0
☐ Farb-Bild	0
☐ Graustufen-Bild	0
☐ Akte	0

OK | Abbrechen

Figure 4: Object relation register – register



Objekttyprelationen

Ordner/ Registertyp

- Geschäftsvorfälle
- Partner-Typ
- Akte

	enthält	Anzahl
☐	Partner-Typ	0
✓	Allg. Korrespondenz	unbegrenzt
✓	Aufträge(Cold)	100
✓	Ausgangsdokument	unbegrenzt
✓	Belege	5
✓	Belege(Cold)	10
✓	Events	unbegrenzt
✓	Farb-Bild	unbegrenzt
✓	Graustufen-Bild	10
☐	Akte	0

Figure 5: Object relation register – documents

Without the object definition the enaio® system is like an empty shell which needs locations to be created for the documents you want to manage. Once an object definition was created, sensible working is possible.

Object types

enaio® distinguishes between the following main types of documents:

Number	Name	Filing tray
00	Folder	In the database
99	Register	In the database
01	Grayscale module (X)	File extensions of the individual pages per document are incremented from 000 to FFF.
02	BW Module (D)	File extensions of the individual pages per document are incremented from 000 to FFF.
03	Color Image Module (P)	File extensions of the individual pages per document are incremented from 000 to FFF.
04	Windows source data (W)	One document is created per file and saved with the corresponding file extension.
05	Video module (m)	File extensions of the individual pages per document are incremented from 000 to FFF.
06	E-mail (Q)	The attachments (multiple pages) are compressed for each document and filed with the file extension 'MIM'.
07	XML module	One document is created per file and saved with the corresponding file extension.

Table 1: Main document type

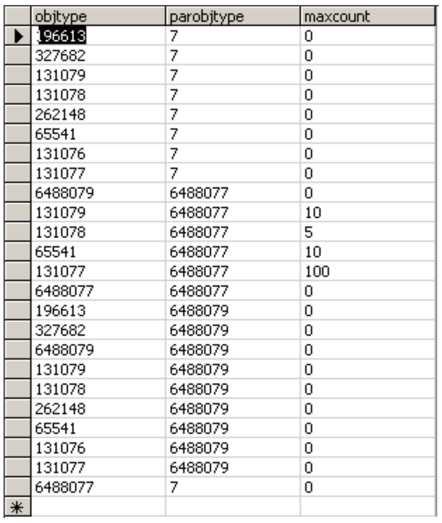
Although folders and registers are not document types, they are listed in the table as the specified main type play a significant role for the administration of DMS objects.

For each intended document main type in each cabinet at least one index form must be created. If you want to archive e.g. file letters in the W-module format `.doc`, a form of the W-module type (main type 04) must be created. Sub types are set up with additional forms. Since the sub type is saved in a WORD format, i.e. 2 Byte data format, exactly 64K (2 Byte = 2^{16}) DMS objects can be created per main type. This applies to documents, folders and registers.

The order in which these forms are created in the object definition dictates the numbering of the document sub types. The first object of a main type receives the sub number 0; the second receives 1 and so on.

The object type is described by a DWORD for which the main type is entered in the HIWORD and the sub type into the LOWORD of the DMS object. The object type is entered into different database tables as it is used to determine the main and sub type of the DMS object (e.g. second created D-Document of a main type in hexadecimal numbers would be 0002 0001: 2 represents the document type, 1 refers to the first additionally created D-Document of the same type, in decimal numbers: 131073).

Another example in Figure 6 shows the table `osobjparrel` in which the object relations displayed in figures 3-5 are saved.



objtype	parobjtype	maxcount
196613	7	0
327682	7	0
131079	7	0
131078	7	0
262148	7	0
65541	7	0
131076	7	0
131077	7	0
6488079	6488077	0
131079	6488077	10
131078	6488077	5
65541	6488077	10
131077	6488077	100
6488077	6488077	0
196613	6488079	0
327682	6488079	0
6488079	6488079	0
131079	6488079	0
131078	6488079	0
262148	6488079	0
65541	6488079	0
131076	6488079	0
131077	6488079	0
6488077	7	0

Figure 6: osobjparrel table with object relations

The object relations are filed in this table with the maximum number of instances (in brackets: the object type in hexadecimal numbers):

- the object type 7 (folder, 'business transactions') cannot contain the following objects:

196613	(0x03 0005)	P-Document
327682	(0x05 0002)	M-Document
131079	(0x02 0007)	D-Document (receipts(Cold))
131078	(0x02 0006)	D-Document (receipts)
262148	(0x04 0004)	W-Document
65541	(0x01 0005)	X-Document.
131076	(0x02 0004)	D-Document (gen. correspondence)
131077	(0x02 0005)	D-Document (orders(Cold))
6488077	(0x63 000D)	Register (receipts) [99=0x63]

- In object type 6488077 (first register "Partner type" in the object definition) the following objects cannot be created, or only in limited numbers:

131079	(0x02 0007)	D-Document (receipts(Cold)) [10 objects allowed]
131078	(0x02 0006)	D-Document (receipts) [5 objects allowed]
65541	(0x01 0005)	X-Document [10 objects allowed]
131077	(0x02 0005)	D-Document (orders(Cold))[100 objects allowed.]
6488077	(0x63 000D)	Register [99=0x63]
6488079	(0x63 000F)	Register (file) (99=0x63)

- in object type 6488079 (second register "file" in the object definition) the following objects cannot be created:

196613	(0x03 0005)	P-Document
327682	(0x05 0002)	M-Document
131079	(0x02 0007)	D-Document (receipts(Cold))
131078	(0x02 0006)	D-Document (receipts)
262148	(0x04 0004)	W-Document
65541	(0x01 0005)	X-Document.
131076	(0x02 0004)	D-Document (gen. correspondence)
131077	(0x02 0005)	D-Document (orders(Cold))
6488079	(0x63 000F)	this register [99=0x63]

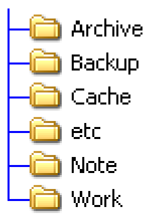
For each DMS object sub type a designated database table is created. The indexing of each object (folder, register, or document) which is created with this index form are saved in this table. In this case folders, registers, and documents are handled differently (# represents a number):

- root# for folder,
- register# for register,
- object# for documents.

A reference via the field `tablename` of the table `osobjdef` is created between the sub type and the name of the table. Each of these tables contains a field `id` in which a unique database index (taken from table `osnextindex`) for each object. This index and the object type (document main and sub type) allow for the exact identification and effective search for objects filed in the enaio® system (documents, folders, registers etc.).

Media Structure

Documents are not filed in a database but in a separate file system. Each server group disposes of its own file system. This file system starts at any path with subdirectories.



WORK

All documents which are currently managed by the system and are not archived are saved in the WORK directory. All application servers access this directory if a document which is not archived is requested by a client program or if a new document is filed (imported) by the system. The path to the WORK area is listed in the database table `path` (consists of Path\\WORK) with flag=0. The WORK area has the following structure in which main type, sub type and document ID (DocID) are processed. Only the last two character of the DocID are indicated for the register at the lowest level.

```
Path\\WORK\\Main_type\\Sub_type\\DocID\\DocID.Page_number
Path\\WORK\\Main_type\\Sub_type\\DocID\\DocID.ext
```

The first notation is valid for all pixel graphics document types (the page number starts with 000), the second notation for W-Documents (e.g. Word), M-, mail and XML documents (`ext` must be replaced with the appropriate file extension, e.g. `doc`). The DocID in the file name is hexadecimal and consists of eight characters (with leading zeros).

In a multi-server architecture as it can be created with balanced servers, all servers must have access to this directory, i.e. this directory must be accessible across the network and entered in UNC notation in the table `path`. A UNC path allows for the direct access (without connection a network drive) of any network resource.

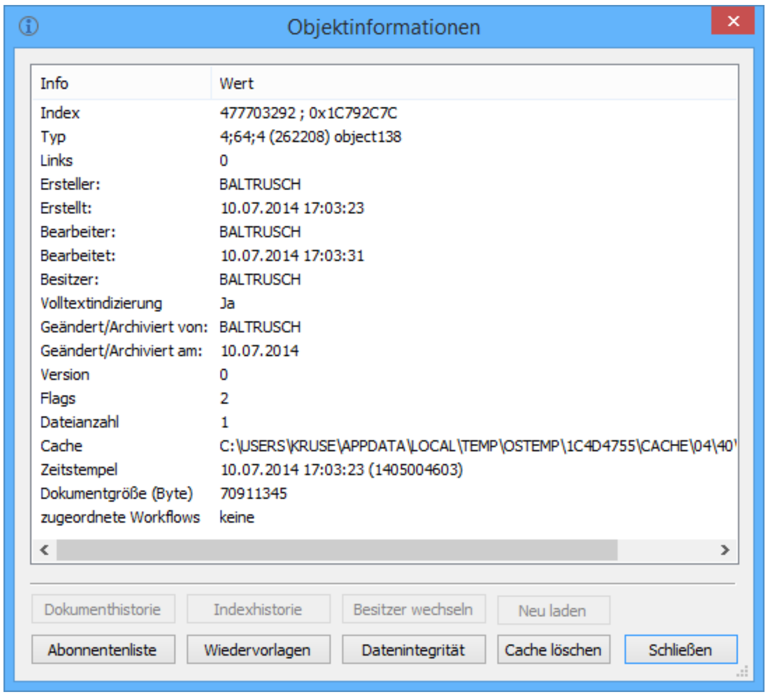
CACHE

If documents are archived the CACHE directory is used for quick file access. The CACHE directory is created by the first launched application server on the same level as the WORK directory (according to the path in the `path` table). An application server first looks in this directory if it contains an archived document. If this is the case, the search is finished and the document is provided. If not, the document is searched on the media listed in the database. All servers of a group require full access to the CACHE directory, too. The structure of the document filing corresponds to the structure of the WORK directory.

If an enaio® system consists of multiple server groups, the CACHE directory also plays a role besides archiving. If a client program wants to access a document, the application server determines the ID of the media on which the document was filed through the field `medium_doc` of the corresponding objekt# table. Then the `server_id` (actually the `ID` of the server group) is determined to which this `media` belongs. This makes it possible to determine if the document belongs to this or a different server group.

- If the document belongs to this group, it is provided and the job is finished.
- If it is a document belonging to a different group, it is differentiated if the document was requested to be viewed or to be modified.
- If the document is supposed to be modified, it is moved to the WORK directory of the server of the respective server group. The entries in `medium_doc`, `medium_name`, `medium_dia`, and `name_dia` of the objekt# table are changed to point to the own WORK area. The server of the other group moves the document and the slide into its CACHE area. Finally, the server provides the document to the requesting client by moving it to the respective client cache.
- If the document will only be displayed, the application first looks for it in its CACHE area. A hash value is used to determine if the document had been modified meanwhile. If this is the current document, it is opened for display. If a document cannot be found in the CACHE area or if it no longer is the current version, it is requested from the server or another group and placed including its slide in the CACHE area. Finally, it is transferred to the client.

This process reduces the data transfer between the server groups which becomes an advantage if only a slow connection between the server groups is available. Please note that the document requested for viewing is not necessarily the current version. Since it is first being looked for in the CACHE directory, it may happen that it contains an older version than available at the owner group. This disadvantage can be compensated. The context menu of enaio® client contains the option 'Object information' (F8). The then opened dialog will display information on the document and the possibility to delete the document from the CACHE area:



Info	Wert
Index	477703292 ; 0x1C792C7C
Typ	4;64;4 (262208) object138
Links	0
Ersteller:	BALTRUSCH
Erstellt:	10.07.2014 17:03:23
Bearbeiter:	BALTRUSCH
Bearbeitet:	10.07.2014 17:03:31
Besitzer:	BALTRUSCH
Volltextindizierung	Ja
Geändert/Archiviert von:	BALTRUSCH
Geändert/Archiviert am:	10.07.2014
Version	0
Flags	2
Dateianzahl	1
Cache	C:\USERS\KRUSE\APPDATA\LOCAL\TEMP\OSTEMP\1C4D4755\CACHE\04\40'
Zeitstempel	10.07.2014 17:03:23 (1405004603)
Dokumentgröße (Byte)	70911345
zugeordnete Workflows	keine

Figure 7: Deleting the CACHE in the 'Object information' dialog

By clicking 'Empty cache' the document will be deleted from the server group cache. A new query will cause the current version to be retrieved from the other server, placed in the CACHE area and forwarded to the client.

ARCHIVE

Next to the WORK directory, also an ARCHIVE directory is created for the archiving process. This directory is used to save the object definitions and all tables of the indexing of the archived documents. A detailed description of the archiving process can be found in the 'Archiving' chapter.

In case of permanent archiving on special media a different filing structure is created on the archive media. The paths to the media can be found in the database table `path` (path\\archive_name) with Flag= '1'. These have the following structure:

```
Path\\Archive_name\\Media_name\\Main_type\\Sub_type\\Document_typ\\DocID\
\DocID.Page_number
```

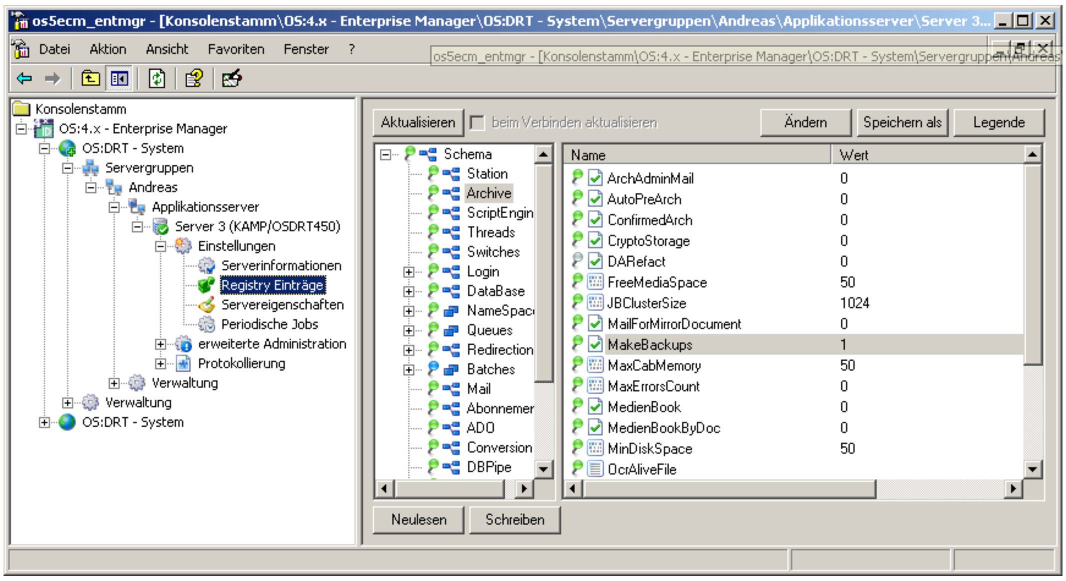
```
Path\\Archive_name\\Media_name\\Main_type\\Sub_type\\Document_typ\\DocID\
\DocID.ext
```

The sub-directory 'Document type' can only have two values:

- '02' for the slide of a document and
- '03' for the actual document.

Backup

To increase data security during archiving, you can edit the configuration in enaio® enterprise-manager to have all documents saved in a BACKUP directory in addition to the archive medium.



Name	Wert
ArchAdminMail	0
AutoPreArch	0
ConfirmedArch	0
CryptoStorage	0
DARefact	0
FreeMediaSpace	50
JBClusterSize	1024
MailForMirrorDocument	0
MakeBackups	1
MaxCabMemory	50
MaxErrorsCount	0
MedienBook	0
MedienBookByDoc	0
MinDiskSpace	50
OcrAliveFile	

Figure 8: Archiving settings

This directory will be created (if needed) by an application server next to the WORK directory. The structure of the document filing location is identical to the structure which is created on the archive media.

NOTE

Next to the WORK directory a NOTE directory which contains document notes is created. Notes are either remarks in text files in which unformatted ASCII text or references to other DMS objects. The text files are managed in the table `remarks`, the references are managed in the table `hyplnk`. Each note contains a system-wide unique ID by which it can be found in the notes file system. The path to the notes (text files) is saved in the table `path` with Flag='2'. The directory structure of notes (text files) consists of main and sub type of the document for which the note was created. Together with the NoteID a path is generated with a structure which is similar to the one of the WORK path:

```
Path\\NOTE\\Main_type\\Sub_type\\NoteID(mod 0x100)\\NoteID.txt
```

When installing a server group, in the table `path` the paths to the WORK (flag=0), the NOTE directory (flag=2) and the later configured paths to the archive media are entered. Here the server groups are allocated (field server_id).

etc

This is different for configuration files. Under optimal_AS 3.60 these were filed in the `etc` directory. It also contained apart from the object definition files (`asobjdef.dat` and `aslisten.dat`) all other configuration files, e.g. `as.cfg`, `asimpexp.cfg`, `ascold.cfg`, etc. Since the preceding version OS:DRT, these files were regarded as resources. To file these files, the directory `etc` is created next to the WORK directory and the files are entered with their entire file names in the table `osresources`.

Application Server

The enaio® server is implemented as an NT service. It provides the business logic in the three-tier architecture of enaio®. The client programs (in the presentation tier) do not access the data directly, but request these from the application server. The server itself identifies the requested information in the database, document tray and document archive (data layer) and makes it available to the requesting client. The client processes the retrieved information (documents, search results,...) and displays it to the user in a sensible manner.

Load Balancing and Scalability

A simple enaio® system consists of a database, an application server, and a number of clients. The more client programs work at the application server or the more complex queries are performed at the application server, the more server capacity is used. Beyond a certain point, the performance of hardware and application server will slow down until queries will require a vast amount of time. Client programs will have to wait and the system will become instable. The application of more resources will solve this problem which occurs in many connected systems. This network property is called scalability. According to requirements, more or less resources are available. The application server of enaio® realizes scalability by 'simply' adding another application server on an additional computer to the DMS and client programs are informed about the further available server connection. This is called 'load balancing' and the servers themselves are called 'balanced servers'. As mentioned before, the client programs are informed about which servers are available in the system. Total load is distributed to all included servers which are now referred to as 'server group'.

All servers of a group must offer the same functions, i.e. they must load the same executors. This condition must be fulfilled as when connecting to a server instance, the offered server functions are still unknown.

During operation, new application servers can be added to the server group to scale the complete system. If the computer has sufficient capacity, you can install more than one application server on it.

The following steps are required to add a new application server:

- Installing a new application server instance in a new server group.
- Informing the clients about the new application server. This is achieved by respective entries in the configuration file `asinit.cfg` on all installed clients.

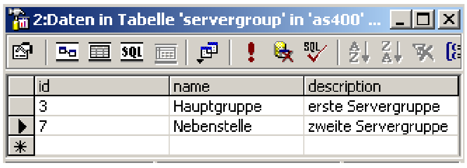
Example of an `asinit.cfg` to connect client programs with multiple servers within the server group 'Hauptgruppe':

```
[Archiv]
Comstring=Hauptgruppe
[Hauptgruppe]
SERVER0=dmsdata1#4010,30
SERVER1=dmsdata2#4020,40
SERVER2=dmsdata3#4030,30
```

A section with the server group name is entered into the entry `Comstring`. This group lists all servers with running application servers and respective port numbers for communication over TCP/IP. Behind this specification, the probability is indicated with which the client connects to this server. In case applied servers are equipped diversely and offer different performances, the probability specification is useful.

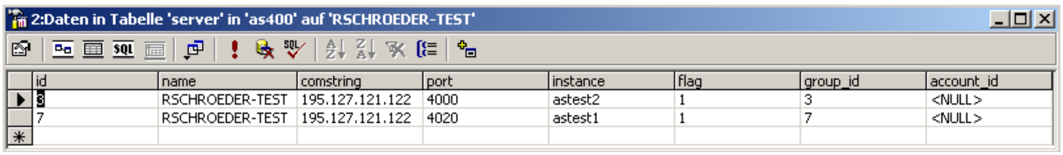
The entry `dmsdata1` is the name of the computer or its IP address in the network. The client connects to this computer and uses the specified port to connect to the application server via a TCP/IP connection.

In the database, server groups and servers are entered into the `servergroup` and `server` tables (see Figure 9 and Figure 10). The WORK and NOTE directories are assigned to the server groups through the tables `path` and `medien` (see Figure 11 and Figure 12).



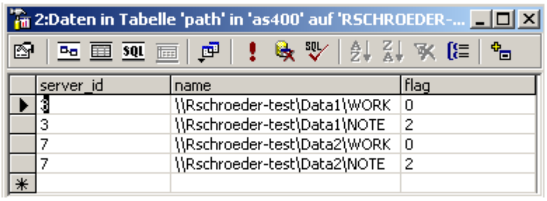
id	name	description
3	Hauptgruppe	erste Servergruppe
7	Nebenstelle	zweite Servergruppe

Figure 9: 'servergroup' table of the database as400 of an enaio® installation.


2:Daten in Tabelle 'server' in 'as400' auf 'RSCHROEDER-TEST'

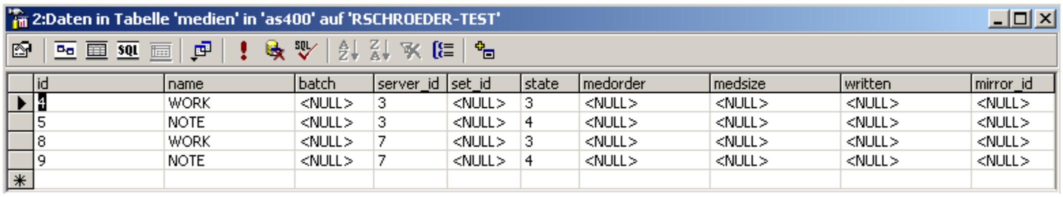
id	name	comstring	port	instance	flag	group_id	account_id
3	RSCHROEDER-TEST	195.127.121.122	4000	astest2	1	3	<NULL>
7	RSCHROEDER-TEST	195.127.121.122	4020	astest1	1	7	<NULL>

Figure 10: 'server' table of the database as400 of an enaio® installation.



server_id	name	flag
3	\\Rschroeder-test\Data1\WORK	0
3	\\Rschroeder-test\Data1\NOTE	2
7	\\Rschroeder-test\Data2\WORK	0
7	\\Rschroeder-test\Data2\NOTE	2

Figure 11: 'path' table of the database as400 of an OS:4x installation.


2:Daten in Tabelle 'medien' in 'as400' auf 'RSCHROEDER-TEST'

id	name	batch	server_id	set_id	state	medorder	medsize	written	mirror_id
4	WORK	<NULL>	3	<NULL>	3	<NULL>	<NULL>	<NULL>	<NULL>
5	NOTE	<NULL>	3	<NULL>	4	<NULL>	<NULL>	<NULL>	<NULL>
8	WORK	<NULL>	7	<NULL>	3	<NULL>	<NULL>	<NULL>	<NULL>
9	NOTE	<NULL>	7	<NULL>	4	<NULL>	<NULL>	<NULL>	<NULL>

Figure 12: 'medien' table of the database as400 of an enaio® installation.

The first server of a group receives the same ID like the server group. All further servers receive the ID in accordance to the status of the table `osnextindex`. Enter the computer name into the field `name`, the IP address of the computer into the field `comstring`, the IP port number the server should use into the field `port`, the service name into the field `instance`. The table in figure 13 shows two server groups on one computer, an improbable scenario in practice but useful for test purposes. The name of the application server from a server group that the client application should connect to is displayed, for example, in the status bar of enaio® client.

With the table `path`, the WORK and NOTE directories as well as the archive media are assigned to the server groups but not to servers; although the field `server_id` indicates so. Each server group has a WORK and a NOTE directory assigned. The corresponding path can be entered into the input field 'Data directory' of the input form 'Database'. Keep in mind that you must enable all servers of a group to access the same data directory. The data directory consequently must be shared. The securest way to do so is to use UNC paths.

The directories and data areas receive individual IDs in the table `medien`. Enter the directories into the respective database tables to realize distributed architectures with more than one server group at different locations.

So documents created in a server group can be made available to client programs that are installed with another IP configuration as they are located anywhere else.

Modularity

enaio® client applications send job requests to the application server kernel that is responsible for job distribution to executive components, system function management and provision of runtime environment. The jobs are queued according to the job names. Each engine is assigned to a queue, i.e. the executor will find its jobs in this queue.

The following engines and namespaces are installed by default:

Engine	Namespace	
axsvckrn.exe	adm	(Administrative jobs).
	lic	(License jobs).
	krn	(Kernel jobs).
oxjobwfm.dll	wfm	(Workflow jobs).
oxjobstd.dll	std	(DMS jobs and archiving).
oxjobocr.dll	ocr	(Character recognition jobs).
oxjobabn.dll	abn	(Subscription function)
oxjobdms.dll	dms	(other DMS jobs)
oxjobcnv.dll	cnv	(Format conversion)
oxjobmng.dll	mng	(System administration job)
oxjobdbp.dll	dbp	(Database piping)
oxjobado.dll	ado	(DB access with ADO)

Table 2: Executors and namespaces

Multithreading

Applications or services are often assigned to the one thread that is connected to the process. The Windows operating system, though, is capable of supporting more than one thread for each application or service. The operating system provides each process (application or service) with limited computing time. After this time has exceeded the current process stops, its status is frozen, and the next process receives time for processing its tasks. The time available for each process does not depend on the number of process running in the thread. For asynchronous tasks, the creation of multiple threads would be very useful. For example, while one of those threads is waiting for data, another thread can perform further tasks with already received information. Waiting for some data does not block the entire process. Precious time would be lost if a server which communicates with a number of clients and waits for database responses would process tasks only in a synchronous manner.

The multi-thread architecture enables you to improve server effectiveness, avoid unproductive queue times for long-term operations, and reduce response times on client queries. Operation that may be accompanied by possible delays (network connection, scheduled tasks, database and file system accesses etc.) will be performed in separate threads.

Though, the number of simultaneously running threads is limited (defined through a configuration parameter) in order to minimize the efforts for switching the thread contexts as well as to avoid unnecessary physical storage occupation.

The server uses the following 7 thread types:

- `HauptThread`

  Transferring configuration parameters, server initialization, initialization of external engines, initialization of other threads, inter-thread communication and synchronization as well as the release of resources and quitting the server.

- `ConnectionThread`

  Establishing a client connection. The thread creates the socket which runs in listen mode, after a successful connect to a new client program the thread creates the necessary structures for the client connection and enters them in the directory of active connections.
- `ReadThread`

  Receiving client queries, syntax check, query integration into the process queues. Each job is integrated into the queue corresponding to the respective namespace. A further thread starts for each 60 client connections a server should run. The maximum number of permitted client connections (MaxConnections) is defined in enaio® enterprise-manager.
- `JobThread`

  Job takeover out of the queue, determining the engine responsible for processing, transferring the client call to the executor for processing, result return to the client program, and triggering error messages. At least one JobThread is connected to each job queue. With configuration parameters you can specify to connect more JobThreads.
- `SendThread`

  Sending notifications (messages) to client programs by use of server events which are not direct results of client queries. Notifications will be sent only to those client programs that have informed the server about the necessity of messages of this type and that have created a specific channel for sending.
- `ConsoleThread`

  Displaying the current state and actions executed by the server in the console window, accepting user commands and forwarding them to threads.
- `ExLoadThread`

  Dynamic change of the executor set running in the server. Every external executor can be loaded into the server at runtime or removed from then server. These actions are also possible in the console (also during remote maintenance).

The multi-thread architecture puts high demands on synchronization and operation coordination of all threads.

Server jobs (queries of client programs or from other servers) are processed in 'queues'. Queues are named, as well. During installation, some standard queues are set up automatically (e.g. common, dms, workflow, dbpipe, ocr). Each queue is provided with one or more Windows threads that jobs of queues are processed in.

The administrator can configure the number of threads for each queue. In one thread a job can be processed. Threads process jobs independently of each other. Job processing in a queue follows the 'fifo' principle (first in, first out), i.e. the first job in a queue is first forwarded to a free thread.

A queue can process jobs of multiple namespaces. Namespaces of engines can be distributed freely and are configured by the administrator after having installed the application server. During enaio® installation default values are entered into the registry.

The following image displayed the internal job process flow in a server:

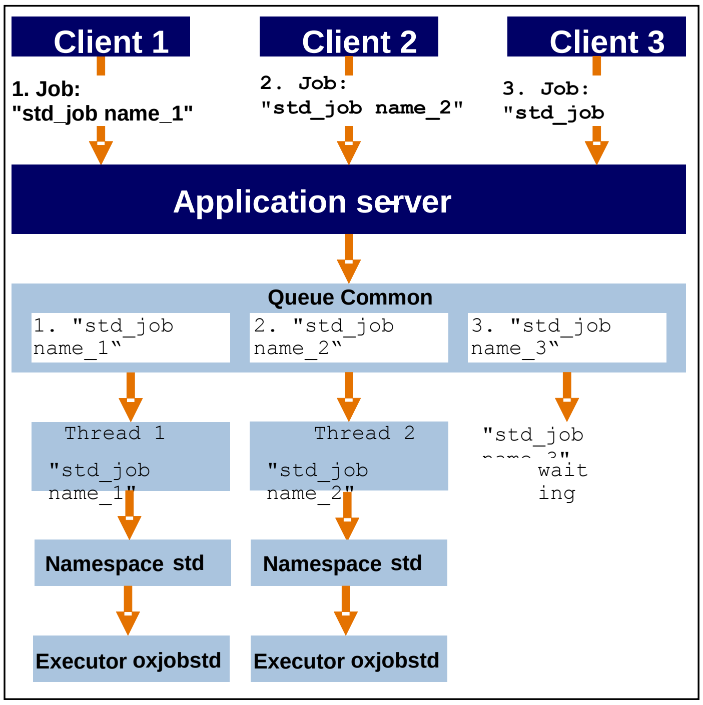


Figure 13: Example for job flow in an application server

In the example, three server jobs are processed that have been forwarded successively from three clients to an application server. The jobs are processed in the namespace 'std'. The executor 'oxjobstd.dll' and the queue common are assigned to that namespace. There are two threads available. As three jobs have been forwarded within a narrow time frame, the first and the second job are transferred to threads but the third job has to wait until a thread becomes free.

Cluster Functionality

The application server can also run on failover clusters. In case of a breakdown a different computer in the failover cluster takes over the application and serves the clients. In the configuration of the operating system (e.g. Windows Server 2008) a virtual IP address is assigned to the cluster which is then resolved to the physical IP addresses of the computers inside the cluster. Which computer is addressed depends on which service is currently activated by the operating system. As opposed to load balancing this makes it possible to specify only one IP address in the connect string - the address of the cluster.

```
[Archiv]
ComString=Cluster#4000
```

The application server services must be set up on each cluster computer under the same service name and the same port. Cluster functionality is achieved by the server services being represented as exactly one server in the system. It important to always have only one service active.

Enter identical computer names for both cluster computers into the registry so they are considered as one system. It may be sensible to use the name of the computer installed first. The entry is made under:

```
HKLM\SOFTWARE\Optimal Systems\ASArchive2\Schemata\4.0\ComputerName
```

This entry (and service name and port number) is used to determine in the database whether or not the server is registered. If there is no entry, the name of the computer the service has started on is used automatically. The difference between cluster and load balancing is there is only one active service in a failover cluster actually making only one service available to the system whereas load balancing includes multiple services or servers accessing the data area. Moreover, with load balancing loads are distributed.

Session Management

Client applications identify a server instance according to the `asinit.cfg` file, connect to it and remain connected during the entire session. When starting a client, a session will be created on an available application server. The client thereby gets a session ID that serves for managing and assigning system resources, e.g. licenses and system files, to a client connection. When terminating the client, either with a job or due to connection loss, the session will be deleted and the used system resources will be unallocated again.

Sessions are managed in the database and entered in the table `ossession`. For each session the server which has initiated it, is saved. Resources being requested and occupied by a session are saved in the table `oslockedres`.

All application servers periodically sign in the database table `ospingtable` at an interval of one minute and note a time stamp of their last ping there. The servers of a group check every five minutes whether or not all other servers did sign in. In case a service does not respond, all sessions connected to this service are flagged in the table `ossession` by setting the field `serverid` to 0.

Simultaneously, a time stamp for the end of the validity of the existing session of the quit server is entered in the column `preservedterm`. The time stamp is calculated as the sum of the current time and a timeout interval of three hours. All servers periodically check if there are expired sessions in the table. If such sessions were found, the connected system resources will be unallocated and the session table entries will be deleted.

All indicated times are default settings and can be changed in the server registry with enaio® enterprise-manager.

Furthermore, when starting a service each server verifies if there are still active sessions that were valid for its ID. In case such sessions are found, the field `serverid` is set to 0 as the physical connection is lost and a new connection must be created.

If a client tries to connect to an expired session ID, it will receive a server error message and has to generate a new session. Since the initial state is restored during automatic release of system resources, all changes made in the meantime are possibly lost.

For DBPIPING all client queries run on the respective application server. Hardware requirements of the computer that an application server is run on are accordingly high.

Client Programs

In accordance with the three-tier architecture, the client programs form the interface with which the user accesses and configures the system, searches for data, and works with documents. enaio® offers four client programs for working with the system:

- enaio® editor to create object definitions,
- enaio® administrator, enaio® enterprise-manager, to configure and administer the system,

- enaio® client, to work with documents and data of the system,
- enaio® capture, to create and process work sequences to import documents.

A detailed description of these modules can be found in the respective handbooks.

System architecture requires only that the client programs can send their queries in the form of XML RPC jobs via TCP/IP to the respectively addressed application server for the session and get a response via TCP/IP as well (Figure 13). Only the server is allowed to access database and file system data directory. Using TCP/IP involves that shared folders are no longer necessary in the network for the client-server communication. This is an important contribution to enhance data security in enaio® systems.

Large networks including many users require frequent and careful maintenance of installed software. enaio® offers the installation of file servers as a continuation of optimal_AS and OS:DRT. File server means that client programs require only one installation on a computer in the system. On other computers network installations are run and the client programs can be accessed over links to the file server. There are only some files and components that must be installed locally. Maintenance of such a system is quite simple. When updating a client program, respective files must be exchanged in the file server computer only.

Integrated Workflow

The product spectrum of enaio® is expanded by the workflow component which is a process-orientated component. It allows the user to visualize and control business processes. The tight link to the DMS offers significant optimization potential for the editing of business processes. Documents are available during all processing steps to authorized users and are automatically forwarded after editing according to the specified process model to the next editor.

The workflow system is discussed in great detail in the system handbook for the Workflow Management System.

Interfaces for other Systems

COM Interface

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. enaio® uses as a real MS Windows application inter-process communication functions provided by the operating system. The hereby implemented COM interfaces are the basis for all further interfaces to third-party systems, e.g. Office Macros or SAP-Link. These interfaces are registered in the usual way and are then available to other applications to communicate with enaio®. A detailed description of the COM interface can be found in the respective handbook.

Office Macros

About enaio® office-utilities

Using enaio® office-utilities gives you access to functions which facilitate your work with W-documents. W-Documents are edited in external applications and thus must be checked out and - after editing - checked back in. Checked-out documents can be opened in the read-only mode by other staff members. As a result, it is ensured that one document cannot be edited by more than one employee at a time.

enaio® office-utilities make checking W-documents in and out easier and allow direct access to enaio® functions from external applications.

The components for data transfer from enaio® – enaio® editor_for Office, Softscanner and PDF-Scanner – are also integrated into enaio® office-utilities.

The enaio® office-utilities functions are available for the following applications:

- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint
- Microsoft Project

Not all functions are currently available for all applications. Version 2003 or later of the Office applications is required for automatic actions.

About enaio® Macros

Data Transfer from enaio®

The data transfer from enaio® adds index data or a document's content to a MS Word document you want to administer and process in enaio®. You can directly transfer data, formulate conditions for data transfer, and modify transferred data.

The data transfer macro replaces the transfer fields you enter into the Word document with respective data.

You can enter the transfer fields and transfer instructions into the document in MS Word or create them with the field editor. The template of a W-application can also contain transfer fields and conditions.

enaio® editor-for-office

Use the field editor (enaio® editor-for-office) to create transfer fields and conditions for data transfer from enaio®. You can enter the transfer fields and transfer conditions into any MS Word document.

enaio® editor-for-office accesses data from the object definition (names of archive objects and index fields) and can only be started at the workstation that enaio® client runs on.

You can start enaio® editor-for-office through the enaio menu or the enaio toolbar. Use the tabs to select field types and to enter required parameters. The field editor therewith creates the syntax for transfer fields.

SAP Interface

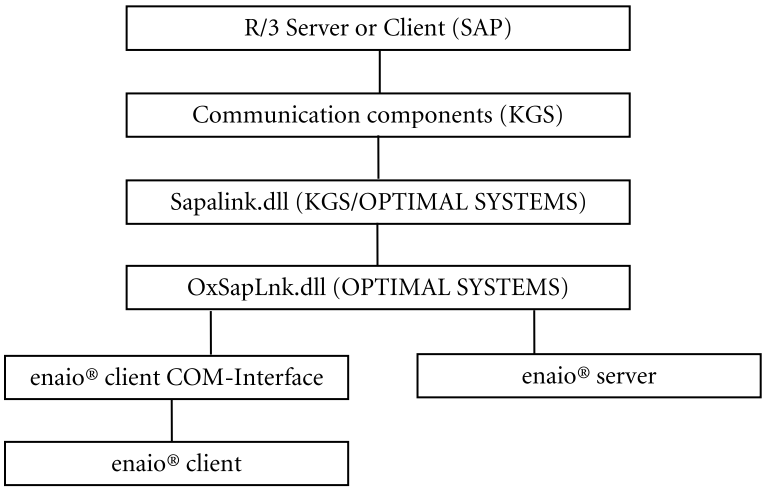
Introduction

The interface enaio®[R/3] connects both system worlds SAP and DMS on data and application level. The interface implements the ArchiveLink standard of the SAP-R/3 system into version 3.1 and 4.5. That allows an already installed SAP R/3 system to be equipped with DMS functions to enable the filing and searching documents as well as index data evaluation. In particular, that may be useful at workstations that cannot only access index data exclusively saved in the SAP system but also to the digital records. Several modules of the KGS Software-Gesellschaft für Systemanalyse und Engineering mbH (software company for system analysis and engineering) have been used for the interface between the SAP world and DMS functionality in order to design the ArchiveLink as version-independent as possible.

System Description

Architecture

All components of the complete system communicate over multiple layers and it was tried to apply a respective module for each logical layer to clearly separate the interfaces from each other. The following figures display the modular connection between single components.


R/3 Server or Client (SAP)
Communication components (KGS)
Sapalink.dll (KGS/OPTIMAL SYSTEMS)
OxSapLnk.dll (OPTIMAL SYSTEMS)
enaio® client COM-Interface
enaio® server
enaio® client

R/3 server or client – according to the application scenario the ArchiveLink of server or client components can be applied. Independently, different functions are launched and used. Especially methods for scanning and display of documents start.

Communication components – represent abstract interfaces between the SAP application layers and the ArchiveLink modules.

Sapalink.dll – interface library that KGS communication components bind in. Out of this library all calls are forwarded to the library oxsaplnk.dll that DMS functions are implemented in.

Oxsaplnk.dll – implementation library of the DMS functionality. Every time a function should run the call is sent as a job to the archive server or DMS client methods for viewing and scanning of documents (components) open.

enaio® client COM-Interface – provides an interface for the OLE automation server to access functions of the client from within other applications.

enaio® server – processing jobs, administering data in the database and managing documents in the file system.

enaio® client – components to view and scan documents

Scenarios and Processes

Server-Side Access

The ArchiveLink is here used to receive, archive, and output large data volumes created within the SAP system, if necessary. In particular, this may involve print lists and ReOrg data. The most important aspect is that server-side access will never run methods for display and scanning of documents, hence, an interface, which in parts asynchronously exits process limits, to the DMS client must neither be created as it nor is suited for server access.

Client-Side Access

With the SAP client documents are mainly accessed for scanning or viewing. Index data are frequently administered through the SAP system but addresses of documents and components are saved in the DMS. When executing a scan method, the interface receives the component's ID that a page should be added to. The scan method is executed synchronously, i.e. the function does not return until the document has been scanned and filed, or an error occurred. That does not apply to the component display as either the internal viewer opens or a respective application starts to display the intended document.

Initialization and Method Editing

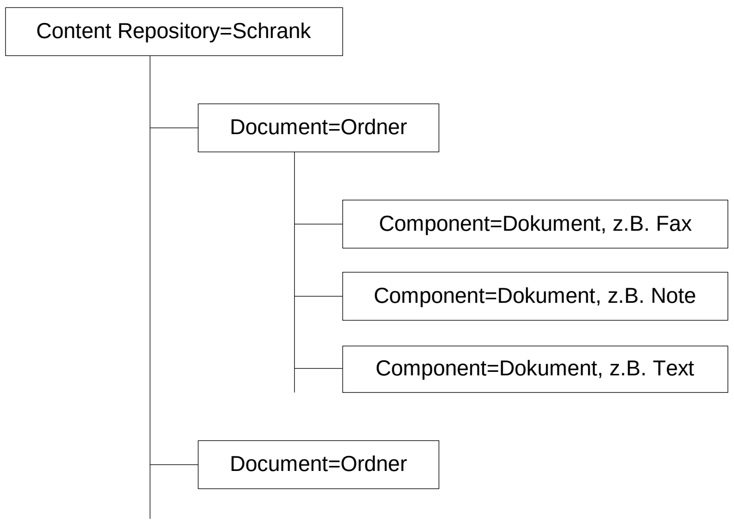
A parameter allows you to start the module oxsaplnk.dll in server or client mode, due to different scenarios. In case the DLL is registered at the server, the client COM interface will not launch and the scan and display methods of components will send an error message. Independently, the COM interface of the client does not launch until a document should be viewed or scanned for the first time.

All method calls neither associated with initialization nor with scanning or viewing of documents will be forwarded to the archive server as jobs for processing. That includes two advantages: for one thing, no connection to the DMS database is required on the ArchiveLink interface and for the other thing, neither the archive server nor the ArchiveLink must run on a computer as long as a TCP/IP network connects both components. The archive server then processes the jobs and returns the results on TCP level. If a file includes the results or the entry of a method, it can either be transferred via TCP or as a link referring to an accessible disk area.

Data Model

To use ArchiveLink, an object definition must be used which was prepared accordingly. For each content repository exactly one cabinet exists in which all generated documents and components and filed.

R/3 - SAPALINK-label	enaio® label
Content Repository (e.g. FI)	Cabinet (e.g. SAP-FI)
Document	Folder
Component	Document (e.g. scanned components, default components)


Content Repository=Schrank
Document=Ordner
Component=Dokument, z.B. Fax
Component=Dokument, z.B. Note
Component=Dokument, z.B. Text
Document=Ordner

SAP documents are also depicted with folders and contain the documents' components. The index data are derived from the attributes of the SAP documents:

SAPALINK-constant	SAPALINK-field meaning	enaio® field	Data type	Comment
docID	SAP document ID	DocID	char(256)	Index field
DOC_PROPERTY_REPOSITORY	Repository to which the document belongs	Content Repository	char(32)	
DOC_PROPERTY_ADATE	Document creation date	Creation Date	char(32)	
DOC_PROPERTY_ATIME	Document creation time	Creation Time	char(32)	
DOC_PROPERTY_MDATE	Document modification date	Date last modification	char(32)	
DOC_PROPERTY_MTIME	Document modification time	Time last modification	char(32)	
DOC_PROPERTY_PROTECTION	Document protection flag	Document protection	char(32)	
DOC_PROPERTY_ALVERSION	Archive Link Version	ArchiveLink Version	char(32)	
	n.v.	Barcode	char(256)	
	n.v.	Barcode sent to R/3	Boolean	

Every component has a content type. Based on this content type it is decided in 'oxsaplnk.dll' and the assignment in the configuration file 'oxsaplnk.cfg' which document type is used in the ECM to apply the contents of the component. If the content type is unknown, i.e. if no entry can be found in the configuration file, the component is assigned to a default W-document type which has no application assigned – this causes the viewer to start the application linked to it under Windows.

All component types have equal index data and contain attributes predefined through the file SAPALINK.DLL. Further index fields are not provided for these document types but should be modeled via other cabinets in order to enable searches in SAP records using the DMS client.

SAPALINK-constant	SAPALINK-field meaning	enaio® field	Data type	Comment
compID	SAP Component-ID	Component _ID	char(256)	Index field
COMP_PROPERTY_FILENAME	original filename	FileName	char(256)	
COMP_PROPERTY_CONTENTTYPE	Contenttype	Content Type	char(256)	
COMP_PROPERTY_ADATE	Component	Creation	char(32)	

	creation date	Date		
COMP_PROPERTY_ATIME	Component creation time	Creation Time	char(32)	
COMP_PROPERTY_MDATE	Component modification date	Date last modificatio n	char(32)	
COMP_PROPERTY_MTIME	Component modification time	Time last modificatio n	char(32)	
COMP_PROPERTY_CHARSET	Charset	Charset	char(32)	
COMP_PROPERTY_ALVERSION	ArchiveLink-Version	ArchiveLink Version	char(32)	
COMP_PROPERTY_VERSION	Application Version	Application Version	char(32)	

enaio® winapp

enaio® winapp allows for the integration of third-party systems that are required to perform enaio® relevant actions. The most important feature is the SAP GUI as it enables searching the enaio® data pool.

When used as a screen scraping software, enaio® winapp can also integrate with other applications: the data extracted from the application is then used to carry out configured actions.

However, it is not possible to return data to the integrated third-party application.

During installation, enaio® winapp is copied to the `\clients\client32` directory. It is configured with the configuration file `axvbWinApp.ini`.

You can integrate conversion libraries that edit and synchronize extracted data in such a way that searches issued against the data pool will produce unique results.

Note that the modules 'OLS' and 'ASC' must be licensed at the performing workstations.

enaio® file-system-archiver

enaio® file-system-archiver allows for passing entire directories and their contents from the file system to the enaio® archive.

The directories and documents are thereby assigned to enaio® objects, or register or document types. Directory and document characteristics can also be saved as index data into enaio®.

During installation, enaio® file-system-archiver, `axfsarch.exe`, is copied to the `\clients\client32` directory. enaio® file-system-archiver requires the 'FSA' license. Special system roles are not required.

For configuration, call the enaio® file-system-archiver executable file with the `-config` parameter. Configuration data is saved to the configuration file `axfsarchconfig.xml` in the application directory. If you want the configuration data instead to be administered in the central `\etc` directory of the data directory, add the parameter `-serverConfig` to the mode parameter when calling. If you use several servers, you have to copy the configuration data into all corresponding etc directories.

You can furthermore add an enaio® file-system-archiver shortcut to the 'Send to...' menu in the context menu. It facilitates passing the desired data to the enaio® system.

Details can be found in the handbook 'enaio® file-system-archiver'.

enaio® communicator

enaio® communicator is a platform allowing for integrating third-party systems with enaio®. As its components access databases, files, and services, with enaio® communicator you can carry out rule-based data import, export, and transformation operations.

enaio® communicator is composed of the kernel and messaging, logging, configuration, and communication components. There are two communication component classes: connectors and transformers.

Connectors

Connectors allow for connecting to external systems:

- periodically (receive -> publish), i.e. the component extracts data and sends them to the internal messaging module.
- event-triggered (subscribe -> send), i.e. as soon as the component receives data it has subscribed to it passes them to an external system.

Thanks to several connectors, data from file systems or databases are accessed via TCP/IP or SOAP, for example.

Connectors are located at the beginning and at the end of communication routes. They offer capabilities for connection status management.

Transformers

Transformers implement special transformation rules.

For example, HL7 and CSV data can be transformed to XML. Furthermore, a transformer can subscribe to and publish several topics (see publish-subscribe).

There are several ways to implement transformers.

- C-DLL
- XSLT
- COM (e.g. in Visual Basic)

Transformers are not meant to access system functions. However, occasionally it cannot be avoided as for rule-based validation operations the archive database is to be queried.

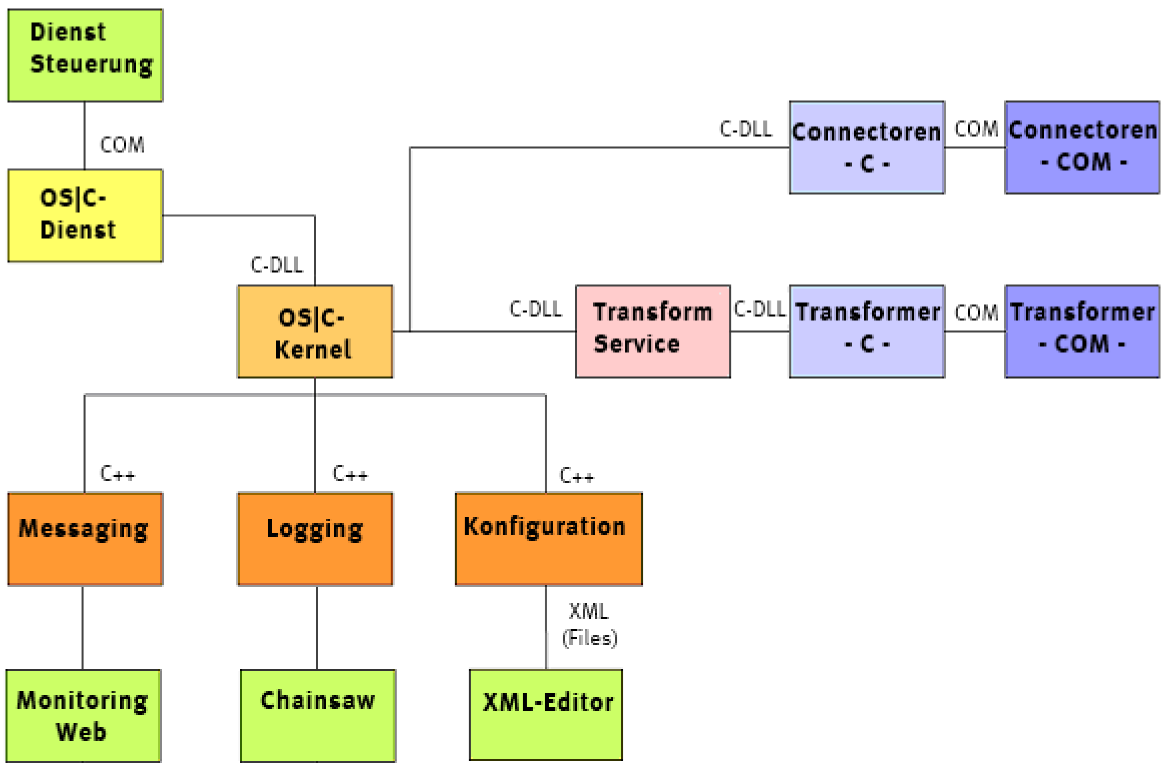
Publish-subscribe

Publish-subscribe systems effectively distribute information and messages to several applications.

A great amount of messages is usually send to interested users (connectors or transformers), thereby passing several servers (enaio® communicator). From this large number of messages, the users subscribe only to the published information they are interested in.

One of this concept's advantages is that the sender and the recipient do not know each other, thereby providing loose coupling between both.

Architecture:



enaio® communicator runs as a Windows service. To run or stop the service, use the Windows service control or the enaio® communicator monitor.

Some components, especially those that use enaio® server functionalities, must be licensed.

enaio® communicator can be installed through the enaio® setup.

Optional Components

In addition to the enaio® core components and interfaces, the number of optional components, which enhance the system by integrating external applications and functions, is continuously increasing.

Viewer Services

Viewer services are core services of enaio® for the flexible display of documents as well as document and index data. Other core services are enaio® fulltext used for full text indexing and enaio® webservice for the integration of external applications with enaio®.

The viewer services can be integrated into enaio® client, enaio® webclient, OS|mobileDMS, or external applications such as Microsoft Outlook.

Viewer services are:

- enaio® documentviewer
- enaio® detailsviewer
- enaio® contentviewer

enaio® documentviewer

enaio® documentviewer integrates the preview of a selected enaio® document into enaio® client as a document preview. The content of a selected folder or register and the preview of a selected object can also be displayed in enaio® webclient. enaio® documentviewer offers simple functions for viewing, navigating, and searching.

Thanks to the integrated enaio® rendition-plus and Rendition Cache components, enaio® documentviewer also enables the conversion of files into other file formats and text recognition in image files.

enaio® documentviewer consists of the following components:

- Web application

  With the Web application, displaying document previews in enaio® client is organized.

- enaio® rendition-plus conversion component

  enaio® rendition-plus generates renditions (images, PDFs, TIFF, text, thumbnails, etc.) from documents and offers text recognition as standard with the integrated CuneiForm OCR engine.

- Storage component Rendition Cache

  Rendition Cache is a cache memory for administering generated renditions centrally. Only one preview is generated per document. If the same document is sent multiple times or a reference document is created, enaio® documentviewer reuses the preview stored in the Rendition Cache.

When you send a document to an internal recipient, enaio® documentviewer allows you to include a reference to the preview file in the email and activate the preview in enaio® webclient. In addition, a thumbnail of the first page of an attached document can be inserted into the e-mail body.

enaio® detailsviewer

enaio® detailsviewer provides flexible HTTP access to index and document data. With the enaio® detailsviewer, for example, index data of marked objects can be displayed in dashlets within enaio® client or in external applications.

enaio® detailsviewer is implemented using the enaio® component enaio® appconnector. enaio® appconnector is a REST interface which can be run as enaio® detailsviewer using a certain configuration. Alternatively, enaio® appconnector can serve as an interface with mobile applications and provide structured access to enaio® server as a communication component. In the latter scenario, enaio® appconnector must be purchased.

The enaio® detailsviewer can be visually adapted for each project. In this case, please contact the consulting team of OPTIMAL SYSTEMS.

enaio® contentviewer

enaio® contentviewer is a framework application that combines enaio® documentviewer and enaio® detailsviewer, enabling a combined view of the two services in enaio® webclient.

For interface projects, enaio® contentviewer can also be integrated differently. Please contact the OPTIMAL SYSTEMS consulting team for information about this.

enaio® appconnector

enaio® appconnector is an enaio® core service.

enaio® appconnector is a REST interface (Representation State Transfer) which provides resource-oriented, flexible HTTP access to index and document data in enaio®.

Thus, enaio® appconnector offers several application areas:

- It serves as an interface, for example, to mobile applications, such as the OS|mobileDMS app for Android-based smartphones, iPhones and tablet PCs.
- It can be deployed as DetailsViewer for index data display in the info window of enaio® client and other, external applications.

With enaio® appconnector, drop targets can be integrated with enaio®.

Drop targets can be used to import data and documents into the enaio® system using enaio® appconnector. With drop targets, various actions in the system can be executed, such as starting a workflow or creating or updating DMS objects.

In addition, the push notification service (PnS) is now an integral part of enaio® appconnector and will be used for the distribution of messages.

The PnS checks regularly whether new messages of subscriptions, follow-ups or workflow process steps are available. If this is the case, the app notifies you of every new message even when it is not active. Users can customize the display of a message in the app.

enaio® appconnector requires you to license the 'APP' module.

enaio® pdfa-dispatcher

enaio® pdfa-dispatcher is used to convert enaio® documents containing image or PDF files to documents containing PDF/A files. The files converted to PDF/A can be inserted into the enaio® archive: you can replace the original file with the PDF/A file, add the PDF/A file as a variant to the original or insert it as a new document.

enaio® pdfa-dispatcher not only allows you to convert image or PDF files of enaio® documents to PDF/A files, it also lets you export the new files.

When additionally running an OCR program during conversion the recognition results are saved as hidden text in the file.

Conversion is either done with the LuraDocument PDF Compressor or the conversion tool integrated with enaio® server.

During the installation of enaio® server the enaio® pdfa-dispatcher components are copied to the server directory. enaio® pdfa-dispatcher is integrated as a service which controls conversion operations on a regular basis.

A component can be integrated into enaio® client as an external program. This enables users to create PDF/A files by sending documents to the enaio® pdfa-dispatcher.

enaio® pdfa-dispatcher requires you to license the 'CLE' module.

TAPI Integration

Thanks to the integration of enaio® TAPI, you can pass phone numbers saved on index forms to the telephone system and have them dialed, or extract phone numbers to issue preconfigured searches to find enaio® objects.

During the installation you can choose to have the components for TAPI integration be set up. The modules 'VBX' and 'VBT' have to be licensed at the processing workstation.

To enable dialing a phone number from within an index form, in enaio® editor you must assign the VB script AddOn (`axvbtapi.vbs`) to a field. The script is installed during the installation process.

Receiving phone numbers is enabled by making the phone number of which incoming and outgoing calls are to be analyzed known to the workstation.

For data protection reasons, this task can be carried out only by users whose accounts have all OS system roles assigned.

To configure searches customize the configuration file `axcstapi-config.xml`.

enaio® mail-archiver

enaio® mail-archiver is an SMTP relay server which can be integrated into enaio® to systematically import e-mails to the archive and start workflow processes based on this data.

Just like any other document in enaio®, e-mails imported with enaio® mail-archiver can be archived in an audit-proof manner.

enaio® mail-archiver not only imports e-mails, it furthermore can be configured to separate attachments from e-mails, save them into enaio®, attach references to the e-mails instead, and forward the e-mails to a Groupware system.

enaio® mail-archiver can be flexibly used in various environments. It can be integrated as an SMTP server, as a POP3 client, or as BCC host.

The components necessary for the installation of enaio® mail-archiver can be found in the `…\disk1\components\OSMAILARCHIVER` directory of your installation data.

enaio® mail-archiver is installed as a service and requires you to license the 'MAR' module.

It is configured with the configuration files `config.xml`, `rules.xml`, and `std-archive.xml`.

Data Security

Support by the System Architecture

TCP/IP is used for the client-server communication. The server opens an IP port for this communication. The file `asinit.cfg` communicates the port number to all client programs. Client programs do never access documents directly. The server makes documents available on client demand using TCP/IP.

That allows enaio® to use a minimum of shared folders in the network. Nevertheless, shared folders cannot be avoided, especially as they are needed for multi-server installations and when multiple server groups are implemented. The file system must be shared if there is more than one server in a group as all servers of a group require access to the group-wide file system with documents. The group-wide configuration (except the archiving) requires the sharing of directories that configuration files are located in for all servers in a group.

But due to the system architecture, there is only one installation architecture with which highest system security possible is achieved in a system:

- that consists of one server group with one enaio® server only,
- that installs the enaio® file system locally on the application server,
- that has the file server for network clients on a second computer in a shared directory. The sharing is required only to enable client access. However, this sharing can be avoided as well if local client installations are used instead of network clients.
- that operates in the DBPIPING mode (the recommended operating mode).

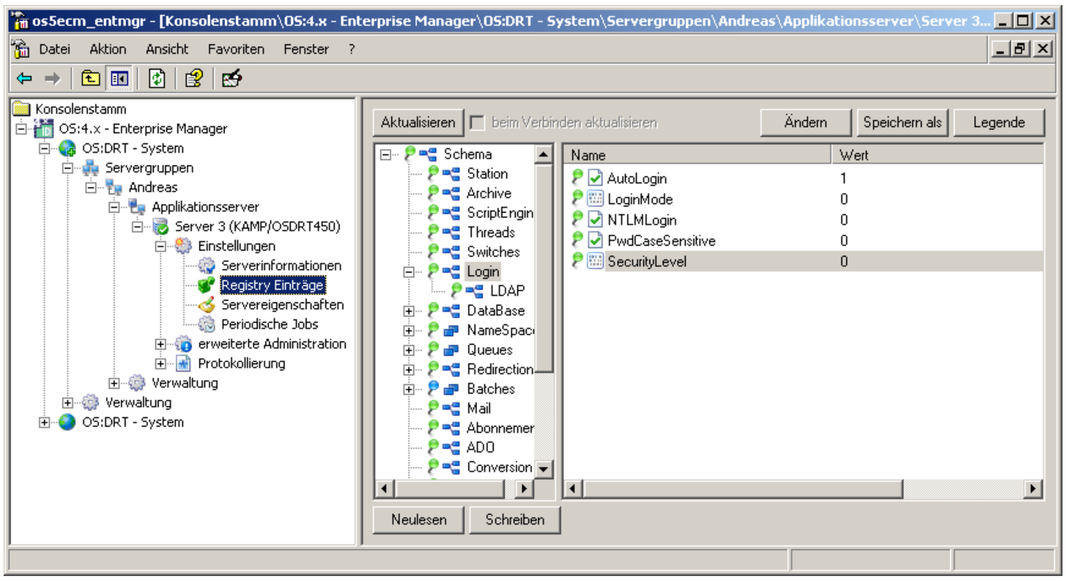
Then, shared folders are not required. The entire communication is performed via TCP/IP, document access is realized on the server without running login sessions.

Users, User Groups and System Rights

Only users that are known in the system can start client programs under enaio®. At configuration, each user receives a password. Users have to enter their passwords into the login dialog that is displayed each time a client program is launched.

To render system abuse almost impossible, three security levels have been integrated that can be configured in enaio® enterprise-manager:

- SecurityLevel 0: only password check (no further limitations)
- SecurityLevel 1: ends the application after three failed login attempts
- SecurityLevel 2: blocks the user account after three failed login attempts

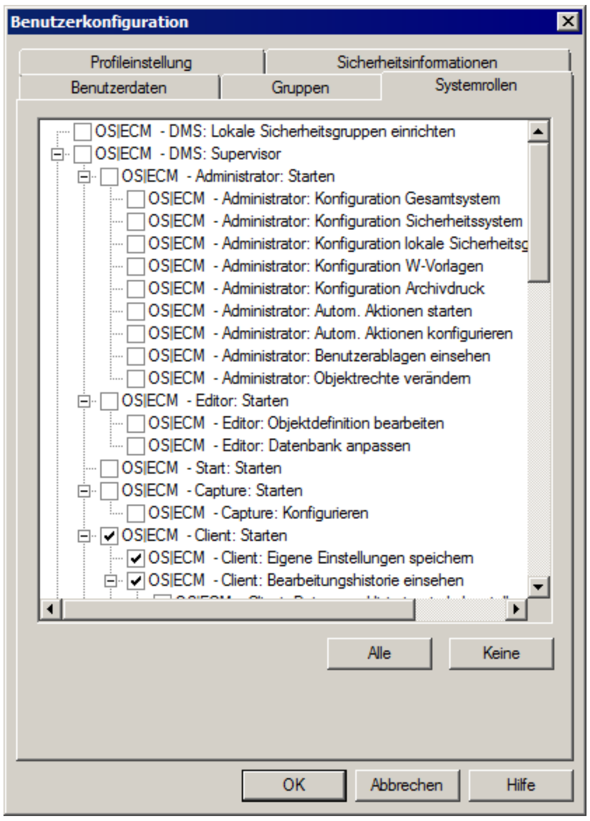


Name	Wert
AutoLogin	1
LoginMode	0
NTLMLogin	0
PwdCaseSensitive	0
SecurityLevel	0

Figure 14: Parameter that specifies the login behavior

Here, the login can be further configured. With the parameter 'AutoLogin' the logged in Windows user can be automatically logged in to the system. With the parameter 'LoginMode' you can choose between login via a dialog box and LDAP login. You can also enabled/disable password case-sensitivity.

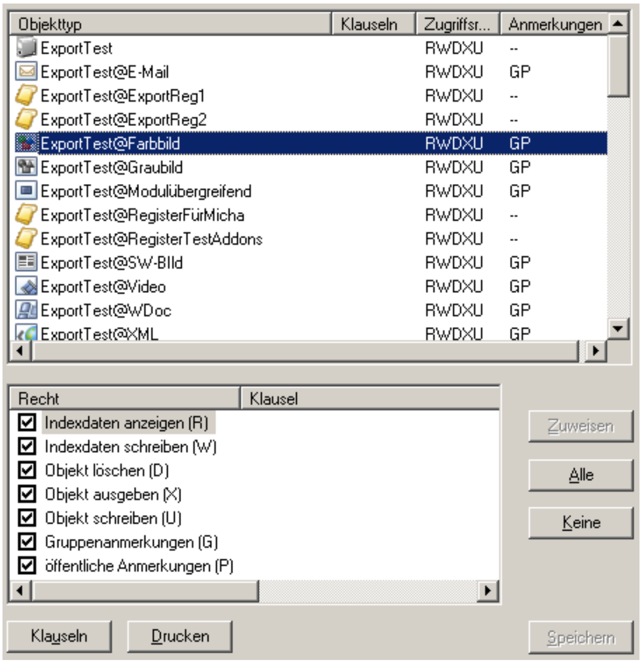
When creating a user account in enaio® administrator, you must assign system roles to the user account in the menu item SECURITY SYSTEM. System roles define the actions a user is allowed to perform under this user account in the enaio® system.


Benutzerkonfiguration

- [ ] OS|ECM - DMS: Lokale Sicherheitsgruppen einrichten
- [ ] OS|ECM - DMS: Supervisor
  - [ ] OS|ECM - Administrator: Starten
    - [ ] OS|ECM - Administrator: Konfiguration Gesamtsystem
    - [ ] OS|ECM - Administrator: Konfiguration Sicherheitssystem
    - [ ] OS|ECM - Administrator: Konfiguration lokale Sicherheitsg
    - [ ] OS|ECM - Administrator: Konfiguration W-Vorlagen
    - [ ] OS|ECM - Administrator: Konfiguration Archivdruck
    - [ ] OS|ECM - Administrator: Autom. Aktionen starten
    - [ ] OS|ECM - Administrator: Autom. Aktionen konfigurieren
    - [ ] OS|ECM - Administrator: Benutzerablagen einsehen
    - [ ] OS|ECM - Administrator: Objektrechte verändern
  - [ ] OS|ECM - Editor: Starten
    - [ ] OS|ECM - Editor: Objektdefinition bearbeiten
    - [ ] OS|ECM - Editor: Datenbank anpassen
  - [ ] OS|ECM - Start: Starten
  - [ ] OS|ECM - Capture: Starten
    - [ ] OS|ECM - Capture: Konfigurieren
  - [✓] OS|ECM - Client: Starten
    - [✓] OS|ECM - Client: Eigene Einstellungen speichern
    - [✓] OS|ECM - Client: Bearbeitungshistorie einsehen

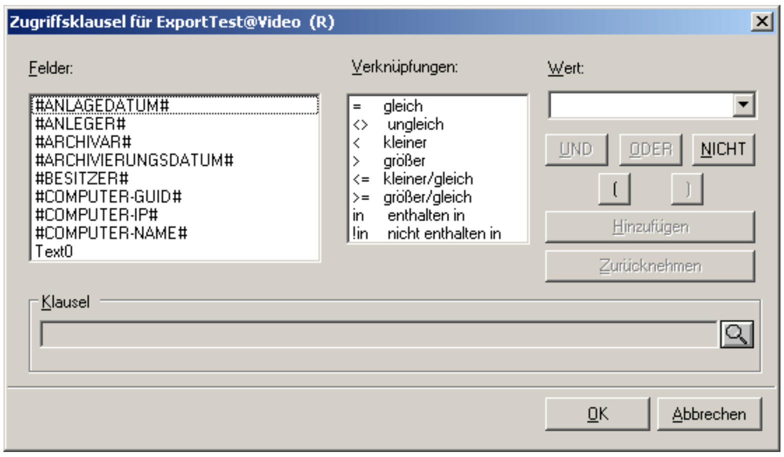
Figure 15: System roles of a user account

In addition to action rights, there are access rights and logical expressions available that define whether or not a user can access DMS objects. You can set up these rights and logical expressions in enaio® administrator for each object type (see Figure 16, Figure 17).


Objekttyp	Klauseln	Zugriffsr...	Anmerkungen
ExportTest		RWDXU	--
ExportTest@E-Mail		RWDXU	GP
ExportTest@ExportReg1		RWDXU	--
ExportTest@ExportReg2		RWDXU	--
ExportTest@Farbbild		RWDXU	GP
ExportTest@Graubild		RWDXU	GP
ExportTest@Modulübergreifend		RWDXU	GP
ExportTest@RegisterFürMicha		RWDXU	--
ExportTest@RegisterTestAddons		RWDXU	--
ExportTest@SW-BIld		RWDXU	GP
ExportTest@Video		RWDXU	GP
ExportTest@WDoc		RWDXU	GP
ExportTest@XML		RWDXU	GP

Recht	Klausel
☑ Indexdaten anzeigen (R)	
☑ Indexdaten schreiben (W)	
☑ Objekt löschen (D)	
☑ Objekt ausgeben (X)	
☑ Objekt schreiben (U)	
☑ Gruppenanmerkungen (G)	
☑ öffentliche Anmerkungen (P)	

Figure 16: Group-dependent rights in the security system


Zugriffsklausel für ExportTest@Video (R)

Felder:

#ANLAGEDATUM#
#ANLEGER#
#ARCHIVAR#
#ARCHIVIERUNGSDATUM#
#BESITZER#
#COMPUTER-GUID#
#COMPUTER-IP#
#COMPUTER-NAME#
Text0

Verknüpfungen:

=	gleich
<>	ungleich
<	kleiner
>	größer
<=	kleiner/gleich
>=	größer/gleich
in	enthalten in
!in	nicht enthalten in

Wert:

Klausel

Figure 17: dialog for access right creation (expressions) to DMS objects

Further Features for Data Security

It is possible to encrypt documents in order to enhance data security and to avoid unauthorized access.

The use of the Microsoft Crypto API, which is a component of the Windows operating system, and the encryption algorithm 'AES' with a key size of 256 bits ensures secure encryption of document files.

Both enaio® client and enaio® server provide the feature of document encryption. Define whether to use one or both enaio® components to encrypt document files since both use different symmetric keys which are part of the program code.

enaio® client encrypts all document files that are sent to enaio® server with the client key and decrypts all document files that are received by the server. All document files are unencrypted before they are saved in the client cache.

Before saving them, enaio® server uses the server key to encrypt document files handed over by enaio® client. If a client requests a document, enaio® server decrypts the document file with the server key before passing it.

Correspondent to latest developments in the digital document area and concerning authorization over chip cards and trust centers, electronic signatures can be added to documents thereby allowing to attest authenticity and to prevent falsification and unauthorized editing.

Amongst others, the license system of enaio® can systematically be applied by users to enhance data security. Unauthorized access to sensitive data and documents in larger networks with many users can be prevented by providing only specific workstations with licenses for encryption or digital signatures.

Installation of enaio®

Overview

This chapter broadly describes installation scenarios of enaio®. A more exact description can be found in the detailed installation handbook of enaio®.

Installations create planned system conditions. The available computer and network technology forms the technological base of the enaio® system with logical client/server architecture. To illustrate logical system properties on the technological base you need to equip multiple computers with enaio® components. There are installations available that install all enaio® components on a computer. However, these are special cases that are realized in productive systems for test purposes at most.

That is why the installation process for the entire DMS usually consists of several single installations at server and workstation computers. Server components must be installed on dedicated server computers and client components on workstations. The installation states on all single computers to be applied in the system determine the status of the entire system.

Single installations include several functions for developing the DMS system:

- development of a productive status at initial installation
- system enhancement
- repair of server or workstation installations
- update or upgrade of servers and workstations within further development of enaio®

enaio® is installed with a setup which was created with Installshield '7'. The administrator who performs the installation is guided by an installation wizard. Decisions in one dialog will lead to different options in the following dialogs. This avoids extensive configuration dialogs which may lead to essential settings being missed out. Instead, exact information is requested to successfully install the system. To reduce the installation effort, additional tools are provided to the administrator, e.g. the *silent setup* or the *maintenance mode*.

On the one hand, the installation status of the enaio® system specifies available installation processes for single computers. For example, client components cannot be installed through network installation until there is one active application server in the system and an available client file server.

On the other hand, the status of the computer that the setup starts on defines installation possibilities. The enaio® components installation depends on whether or not components have already been installed on this system. During the installation of enaio® components information on the installation are saved in the Windows registry. So the maintenance mode of InstallShield becomes available that simplifies enhancement, update, and repair of the installation. Maintenance mode is only available if enaio® has already been installed on this workstation and if it neither has been accidentally nor explicitly disabled. But the maintenance mode is limited as you cannot install an application server in this mode. If necessary, you can disable maintenance mode.

Installation scenarios depend on system planning. Afterwards, several enaio® system components must be installed according to the plan on selected computers in the network.

An enaio® system can consist of the following software components:

- application server,
- various enaio® clients,
- enaio® web components,
- other enaio® components, e.g. Office macros etc.,
- non-OS component, e.g. database, hardware integration etc.

Although the database is an essential component to run the enaio® system, it is not a part of it. Setup, maintenance and backup of the database are not part of the administrative tasks within the enaio® system, despite the importance of these tasks. Hardware drivers for scanners, jukeboxes etc. are also not part of the enaio® system.

You have to distribute these components on all network computers in a way that allows you to use full system performance of enaio®. Due to the high flexibility of enaio® numerous system variants and different system concepts can be realized. You can either install a system with all components on a single workstation or set up simple or complex system. These three installation models and necessary setup steps will be described exemplary.

Antivirus protection has an influence on applications which frequently access the file system. This concerns particularly memories and caches of the enaio® server, databases and the e-mail components. It is considered in enaio® that memories and caches are only read and described by enaio® components. Antivirus protection has therefore to be coordinated with the surrounding application components and, if necessary, single directories will have to be excluded from scan processes.

Expansion Levels of the enaio® System

Due to the high flexibility of enaio® and its installation routines numerous system variants and different system dimensions can be realized. Thereby, it does not matter of you want to set up a single workstation system, or complex systems with several network segments. Three available expansion levels are briefly presented here: a single workstation system which is rarely found in practice, a typical installation with an application server, and a complex system.

Single Workstation System

For such system all components are installed on one computer. This is not a typical enaio® system and is mostly used only as a test system. The following figure shows such a system and the installed components.

Figure 18: Single workstation system

Installation requirements of the enaio® system when installed on a single computer are equal to requirements of complex system installations: licenses are required and hardware and software must be suitable.

On this computer, the freely available MSDE database, the application server and the administrative and productive clients are run. Therefore, the computer must meet the requirements for each of these installed components except that a server operating system is not necessarily required.

In case of a single workstation system, the entire installation can be performed in one single step. To do so, choose the installation option 'User defined'. Then all required components will be installed.

Once the installation was completed, the required administrative tasks to start the operation of the system are carried out.

Simple System with one Application Server

This is a typical enaio® system. The components are distributed across multiple computers which are connected through a TCP/IP network. The following figure shows such a system and the installed components.

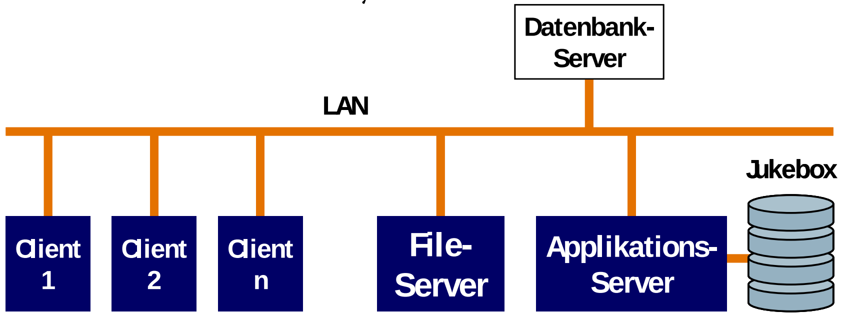


Figure 19: Simple system with one application server

There are several installation requirements for the system as client components need other conditions than server components. In a network an adequate database server offers its service independently.

A simple system with one application server and a file server requires the following steps:

- The database must become accessible first.
- Then the application server is installed.
- Then the file server is installed.
- Finally, all required clients are installed by performing a network installation.

Once the installation was completed, the required administrative tasks to start the operation of the system are carried out.

Complex System with more than one Server Group

This is a highly expanded enaio® system. The components are distributed across multiple computers which are connected through a TCP/IP network. The complete system is distributed across multiple network segments which not closely located. Some clients may even be connected via RDT. The system consists of the following components:

- A number of application servers with multiple jukeboxes, balanced servers and server clusters
- multiple server groups
- multiple file servers
- a large number of installed clients with installed scanners
- a number of enaio® capture clients with multiple high-performance scanners
- installed enaio® web

The following figure shows such a system and the installed components.

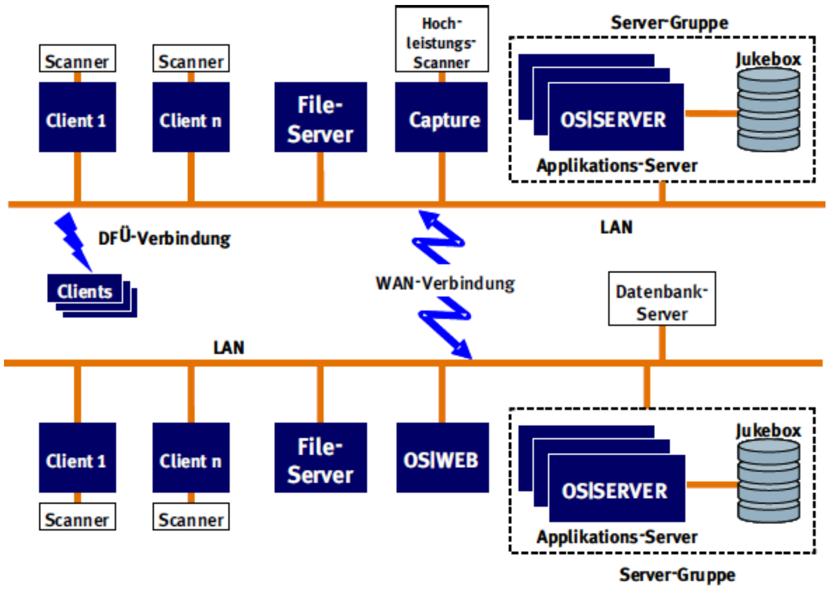


Figure 20: Complex enaio® system

Depending on the complexity of the system, different installation steps will have to be performed. Most steps must be repeated in systems with multiple application servers for each server group. The procedure could be as follows:

- First, the application server is installed in the first server group. The required jukebox drivers must be installed.
- The intended number of application servers is added to the first server group
- Further server groups are added
- The intended number of application servers is added to each server group
- Then the file servers are installed
- All required clients are installed. The required scanner drivers must be installed.
- All required enaio® capture clients are installed.
- enaio® web is installed

For all installation steps, a suitable and available database is required.

The order of the individual installation steps may differ from system to system. Sticking to the described order is not necessarily the fastest way to set up a system. In any case, the application server must be installed first. Further dependencies exist. E.g. a network installation can only be executed if a file server exists and the file server directory is shared.

To reduce the installation time, the system can be configured while the components required by the destination system are being installed. As soon as an application server and an administrative client have been installed, an object definition can be designed in enaio® editor. Other required configurations can be performed in enaio® administrator.

Creating Documents

General Requirements

To create folders, registers and documents the client requires an object definition. The following object definitions can be created and defined in enaio® editor:

- the object types in the DMS
- the dialog elements of the index form for the object types
- additional modules for document types used to create corresponding documents
- and the object relations which are used to define on which hierarchical level of a folder type which register and document types can be created

enaio® editor also creates the database tables adapted to the object definition in which the following data are saved:

- Indexing
- further automatically saved data, e.g. 'creator' and 'creation date' – basic parameters
- data for the editing history
- object relations

At startup the client loads the object definition and the group-specific access permissions to DMS objects from the database, determines the permissions of the logged-in user and provides the functions to access and to create DMS object.

The following data are administered using enaio® administrator and stored in the database:

- Users
- assignment of users and groups
- group-specific access rights for archive object types

If a user creates a new DMS object in enaio® client, the client will open the index form of the selected object type. The user saves the indexing and the client enters the data, indexing, location and a unique ID which it receives from the server in the database.

Documents which have files assigned by users are locked by an entry in the database. Other users cannot add data during this time, but the indexing can be viewed and edited by more than one user.

Then the user assigns files to documents. For each type of document, e.g. image documents, Windows documents, e-mails, video documents, and XML documents, this process differs.

The client manages the document files in a user-specific WORK and CACHE area. If a user files document files, they are transferred to the server and saved by the server in the WORK area.

The Object Definition in the Database

The object definition is created in enaio® editor and saved in the database:

- the table `osobjdef` contains the definition of the DMS object types
- the table `osobjfields` contains the definition of the individual index fields of the DMS object types
- the table `osconf` contains, amongst others configuration entries of index fields, catalog data
- the table `osproperties` also contains the properties of DMS object types

The table `osobjdef` contains in the column `tablename` the name of the table in which the indexing of an object of this type is saved.

Object Relations in the Database

Via the object relations you can determine on which hierarchical level of a folder type a register or document type can be created. Object relations are created in enaio® editor and saved in the table `osobjparrel`.

The table has the following structure:

Column	Contents
objtype	the object type of a folder register type
parobjtype	the object type of the objects (folders/registers), in which restrictions with regards to the instances of 'objtype' apply
maxcount	Maximum number of objects to be created of the type

Table 3

Object relations are not taken into account during search and display of lists of contents.

Configuration Entries for Catalogs, AddOns and the Dialog Element 'Table'

Catalogs and AddOns can be assigned to dialog elements in enaio® editor. Catalogs are simple or structured lists from which users can select entries for the indexing. AddOns are libraries providing additional functions, e.g. the structured indexing function.

The table `osobjfields` with the definition of the index fields contains the column `flags`, which contains the information if and which catalog or AddOn is assigned.

Configurations are necessary for many catalogs and AddOns. In such case an ID for the configuration data is entered in the column `osconfid` of the table `osobjfields`. The table `osconf` contains the configuration data in a BLOB field for each `osconfid` in the column `osconfval` in the same way it is displayed in enaio® editor. The `osconftyp` type of catalog and AddOn entries in the table `osconf` is `6`.

AddOns are saved without path, they are searched for in the directory `\clients\client32`. Structure tree files, external programs and scripts are also looked for in the same location. If they cannot be found there, they are searched for at the path with the mapped drive stored in the table `osconf` in the column `osconfval`.

Configuration information, i.e. column names and column properties, are managed for the dialog element 'table'. As for catalogs and AddOn, an ID for the configuration data is entered in the table `osobjfields`. The data are managed in the table `oslistctrl`. For each column it contains the name in the client, data type, field length, and column name in which the indexing is entered in the table `object#list#`. (# is a consecutive number)

Indexing of Folder, Register and Document Type

enaio® editor creates for each DMS object type an indexing table when adapting the tables. The table contains for each created index field a column and further columns for system data and basic parameters. The location of the DMS objects within the hierarchical structure of folders and registers is saved in the location tables.

Folder type tables are labeled with `stamm#`, register type tables are labeled with `register#` and document type tables are labeled with `object#`.

The table names are displayed in the properties dialog of the DMS objects in enaio® editor, the field labels are displayed in the database view of enaio® editor.

The tables `stamm#` (folder) have the following structure:

Column	Contents
`id`	the unique ID of the DMS object
`time stamp`	the exact time when a DMS object was filed
`left`	flag, if notes exist
`foreignid`	the ID of another archiving system
`systemid`	the unique ID in another archiving system
`creator`	the user who created the object
`created`	the time of creation
`modifyuser`	The user who last modified the object
`modifytime`	the time of the last modification
`deleted`	delete flag for the object
`osowner`	owner of the object
`field#`	the indexing of the DMS object in the individual fields

Table 4

The names of the columns for the dialog elements are displayed in the database view in enaio® editor and are also entered in the column `fieldname` of the table `osobjfields`. The field labels predefined by enaio® editor correspond to the data types. The editor uses `feld`, `real`, `datum`, `zahl` and `list` for the dialog element 'table'.

The tables `register#` (register) have the following structure:

Column	Contents
`id`	the unique ID of the register
`stamm_id`	the ID of the folder type which corresponds to the register
`parent_id`	the unique ID of the register if a register is inside another register.
`foreignid`	the ID of another archiving system
`systemid`	the unique ID in another archiving system
`left`	flag, if notes exist
`creator`	the user who created the register
`created`	the time of creation
`modifyuser`	the user who last modified the register
`modifytime`	the time of the last modification
`time stamp`	the exact time when the register was created

`deleted`	delete flag for the register
`osowner`	owner of the register
`field#`	the indexing of the register in the individual fields

Table 5

The tables `object#` (document) have the following structure:

Column	Contents
`id`	the unique ID of the document
`time stamp`	the exact time when the document was created
`haupttyp`	the main type indicates the module of the document type
`untertyp`	the subtype indicates the document type, the unique object type can be derived from the main type and sub type
`anzahl`	the number of files of a document
`created`	the time of creation
`creator`	the user who created the document
`archived`	the archiving date for the object information
`archivist`	the archivist for the object information
`flags`	indicator of the archiving status
`left`	flag, if notes exist
`medium_doc`	the ID of the section in which the document is saved
`name_doc`	the path and the name of an archived document
`medium_dia`	the ID of the section in which the slide of the document is saved
`name_dia`	the path and the name of an archived slide
`version`	names of variants for Windows documents
`lockuser`	the user who has checked out the document
`foreignid`	the name of another archiving system
`systemid`	the unique ID in another archiving system
`modyfyuser`	The user who last modified the object
`modyfytime`	the time of the last modification
`deleted`	delete flag of the document
`osowner`	owner of the document
`field#`	the indexing of the object in the individual fields

Table 6

An additional table, a shadow table, is created for each folder, register and document type. These tables have the same names but with an appended 's'. If in enaio® editor the property 'Create index data history' was assigned to a DMS object type, the entire existing indexing is transferred into the shadow table.

The shadow tables have the following structure:

Column	Contents
`osguid`	the unique ID of the change
`field#`	the indexing of the object in the individual fields before the change

Table 7

All data of the change, for example time, user, workstation, type of change, are entered in the table `osobjhist` and are also labeled with the unique ID there.

If a document type had the property 'Create document history' assigned, enaio® server will save the documents before the change in a sub-directory in the WORK area. The subdirectory is named after the object ID (hexadecimal).

The indexing of the dialog element 'table'

The indexing of the dialog element 'table' is saved in designated tables created by enaio® editor, just as multi-fields. The table names have the following structure: `object#list#`. Example: the index field `list3` of the document type `object27` is a table. The database table is named `object27list3`.

The database tables have the following structure:

Column	Contents
`id`	the unique ID of the DMS object
`line`	the row number
`field#`	the indexing in column no. #

Table 8

Along with the tables for database tables, shadow tables for index data history are created. Their name will have the character 's' appended. The column names in the client and the properties of the columns are saved in the table `oslistctrl`.

Register and Document Location

Documents are saved in enaio® in a hierarchical structure of folders and registers. Folders make up the top level. Registers and documents can be created in folders, further registers and documents can be created within registers. Registers are always located within a folder or another register, documents are also located within a folder or a register. The location of a register is stored in the table `ossparregrel` and for documents in the table `sdrel`.

Register Location

The location of a register is stored in the table `ossparregrel`.

The table has the following structure:

Column	Contents
`folderid`	the unique ID of the folder in which the register is located
`foldertype`	the ID of the folder type in which the register is located
`registerid`	the unique ID of the register
`registertyp`	the ID of the register type
`parentregid`	the unique ID of the register of the next higher level
`parentregtyp`	the ID of the register type of the next higher level

Table 9

Document Location

The location of a document within the archive is stored in the table `sdrel`.

The table has the following structure:

Column	Contents
`loeschen`	currently not used
`zeitstempel`	the exact time when a DMS object was filed
`stamm_id`	the unique ID of the folder in which the document is located
`object_id`	the unique ID of the document

`objekttyp`	ID of the document type
`register`	the unique ID of the register of the next higher level
`regtyp`	the ID of the register type of the next higher level

Table 10

The filing location of the document files is not stored in the database. It derives from the sub type and main type and the ID of the document.

Logging in the Database

Every action is logged by the client in the table `osobjhist`.
The table has the following structure:

Column	Contents
`osguid`	the unique ID of the action
`osid`	the unique ID of the DMS object the action refers to
`ostype`	object type ID
`osuser`	the unique ID of the user who has executed the action
`ostime`	the time stamp
`osaction`	an action ID
`osstation`	the GUID of the workstation on which the action was executed
`osinfo`	the info text which is displayed in the editing history next to the action description

Table 11

Logging of these actions is independent of OXRPT logging which can be configured for all components of enaio®. A label and a detailed description which is displayed in the editing history is assigned to the action IDs in the table `oshistact`. Below you will find an incomplete overview of action types which are logged and displayed in the editing history:

osaction	**osinfo (label)**	**osextinfo (detailed description)**
1	Digital signature	The document was signed digitally. The digital signature can be verified via the view options.
2	Object created	The object was created by functions of the client or by import.
3	Index data modified	The index data of the object or its status was edited by functions of the client or by an update during import.
4	Document modified	The document was edited by functions of the client or by an update during import.
5	Document archived	The document was archived audit-proof. Changes are no longer possible.
6	Document deleted	-
7	Document output	The document was read by the user, printer or issued otherwise. There was no editing.
8	Document status changed	The document's status was set to archivable.
9	Document status	The document's status was set to not

	changed	archivable.
10	Document created	The document was created by functions of the client or by import.
11	Reference created	-
12	Reference removed	-

Table 12

Actions 3 (Index data modified), 4 (Document modified), 11 (Reference created) and 12 (Reference removed) are stored as modification data.

Locking Documents in the Database

If a user creates a document, it is first indexed. The client writes the data into the table `object#` and enters the location into the table `sdrel`. The client enters the document also into the table `doclock`. Thus no other user can associate document files with it. The locking mechanism is activated each time a user edits a document. This is indicated in enaio® client by a lock icon. Once the user has finished editing the document, the document is removed from the table `doclock` and is released for editing.

The table `doclock` has the following structure:

Column	Contents
`object_id`	the unique ID of the DMS object
`object_type`	object type ID
`user_id`	the unique ID of the user who edits the document
`user_time`	the time stamp
`info`	status information
`fileinfo`	modification date of Windows Documents
`station`	user workstation

Table 13

Indexing and Location of DMS Objects

Folders, registers and documents are created as follows:

- Data import
- enaio® capture
- enaio® client

enaio® client loads the object definition, the object relations, and the catalog, AddOn, and table configuration entries from the database at startup.

If object definitions, object relations and configuration entries are changed after enaio® client was started, enaio® client must be restarted to receive the current data. All enaio® server instances must be restarted too, or the affected executors ('std' and 'dms') must be reloaded for all servers.

enaio® client determines at startup the rights of the logged on user according to the access rights and group assignments in the database.

If these data are changed after the start of enaio® client, enaio® client can update these data via Shift-Ctrl-F5.

enaio® client also loads user-specific settings from the table `osconf`. In this table user-specific settings are stored for every user (column `osconfuser`) as type 3 (column `osconftype`) in the column `osconfval`. These and other settings are loaded at startup and saved upon exiting the client.

Not user-specific settings are managed in the file `as.cfg`. The path to this file is saved in the table `osresources`. When starting the client, the server creates a copy of the file `as.cfg` in the directory

`\Temp\ostemp\'ServerName'#'Port'\etc\.`

The client loads the settings from this file.

If these data are changed after enaio® client was started, enaio® client must be restarted to receive the current data. The system administrator can use enaio® enterprise-manager to force any arbitrary or even all clients to quit.

Folders, registers and documents are indexed and have a location inside the DMS assigned. The indexing is entered by the user manually or by functions inside the type-specific index form which is created by enaio® client from the data it loads at startup. The client verifies the data if the user saves indexing data. Data which do not meet the specifications are flagged accordingly and the user is requested to correct the entries. If the indexing matches the specifications, the data are saved in the database.

AddOns can have the property 'Execute automatically'. They will be automatically executed once the indexing is saved.

The location of the folder is always the top level of the hierarchical structure. If a user creates registers or documents, he first has to open a folder by which he determines the location. Documents can also be stored in a user-specific filing tray. The user can select the filing tray or an already open folder as a location for documents which are transferred from the file system.

Caching index data

Users can cache the last entered indexing for every object type and have it entered into a new index form. These data are saved along with other user-specific settings in the table `osconf` which is loaded at startup and saved upon exiting the client.

The indexing of the dialog element 'table' is not cached. The cached data is not verified if it matches the specifications when being entered. The data are saved according to the tab order of the fields. If the order is changed in enaio® editor, cached data are entered in the wrong fields.

Creating Folders and Registers

If a user creates a new folder, the client will open the index form. Once the user saves the data, the client will request an ID for the folder from the server. The server determines the next available ID in the table `osnextindex` and enters the assigned ID, its server group ID and the time stamp into this table and sends the ID to the client. The client enters the data in the corresponding table `stamm#` and logs the action in the table `osobjhist`.

If a user creates a new register, the client enters the location of the register into the table `ossparregrel`.

Example of the table structure after creating a register:

The server enters the last signed ID into the table `osnextindex`:

id	serverid	time
1311	2	1033551104

The client enters the data into the table `register#`:

id	stamm_id	parent_id	feld1
1311	1203	1286	Indexing

The client enters the location into the table `osparregrel`:

folderid	foldertype	registerid	registertyp	parentregid	parentregtyp
1203	22	1311	6488079	1286	6488079

The action is logged by the client in the table `osobjhist`.

Creating Image Documents

Image documents are documents which can have grayscale images, black/white images or color images assigned. Which file type can be assigned to which document type is specified while setting up the document type in enaio® editor. This type is the main type of a document and dictates the editing module of the documents and, thus, the file format. Image files are not identified through file extensions in the DMS but through the main type.

Main type	Image document type	Module extension	File format
1	Grayscale Images	X	JPEG (maximum quality)
2	Black/White images	D	TIFF G4
3	Color images	P	JPEG (maximum quality)
4	Windows	W	Application format
5	Video documents	M	MPEG, AVI
6	E-mail	E	MIME, MSG
7	XML		XML, XSL, XSLT

Table 14

The editing module for grayscale images, black/white images, and color images in enaio® client process data through the TWAIN or KOFAX interface from scanners or directly from the file system and save them in the DMS format in the workspace. They are named `AS'Module_extension'`.

Image files with the following formats can be transferred from the file system:

- TIFF files (`*.TIF`) - uncompressed or Fax G4,
- JPEG files (`*.JPG`),
- BMP files (`*.BMP`),
- PCX files (`*.PCX`),
- Dicom files (`*.DCM`),
- Targa files (`*.TGA`),
- Gif files (`*.GIF`) - uncompressed only.

The color format will be converted, if necessary, and the files are saved in the DMS file format. The module for grayscale images converts color images into grayscale images, the module for black/white images converts color images and grayscale images into black/white images.

Image Files on the Client

If the user has selected all files for the pages of the image document and files a document, the client saves the files in the user-specific cache.

Path and file name have the following structure:

```
..\[user id]\CACHE\[main type]\[sub type]\[mod 0x100]\[DocID].[page]
```

`[benutzerid]` is a user-specific directory which is labeled with the hex ID of the user.

`[Haupttyp]` is the hexadecimal main type of the document.

`[Untertyp]` is the hexadecimal sub type of the document,

`[mod 0x100]` corresponds to the last two digits of the hexadecimal document ID,

`[DokID]` is the hexadecimal document ID,

`[Seite]` is the page number, incremented hexadecimal starting with `000`.

If the document type has the property 'Create slides', the client will also create quicklooks, small previews of the first page. The quicklook file of a document is saved in the same directory as the image file and has the same name. Its extension is `DIA`, the image format is TIF.

Example for the path and file name of an image file:

`\Temp\ostemp\000000ED\CACHE\02\1B\36\00000136.001`

`\Temp\ostemp` is the workspace, `\000000ED\CACHE` is the CACHE area of the user with the ID 237, `02` is the main type (black/white image), `1B` is the sub type of the document type, `36` are the last two digits of the document ID, `00000136` is the unique document ID of the document, `.001` is the file extension for the second page of the document.

Image Files on the Server

Once the client has filed the document, it transfers image files and the quicklook file to the server and deletes the files from the WORK and the cache area. The server saves the image files and the quicklook files in the WORK area.

Path and file name in the WORK directory have the same structure as in the cache:

```
\WORK\[main type]\[sub type]\[mod 0x100]\[DocID].[page]
```

Table Structure when creating an Image Document

Similarly to the creation of registers, the client obtains an ID from the server of the document and enters the indexing in the table `object#`. If the user has not yet filed the document, his user ID is entered in the column `lockuser`. The client enters the document also into the table `doclock`. At this time the document can be searched by other users, although it has no pages yet and is locked for editing. The indexing is not locked and can be edited.

The client enters the document location in the table `sdrel`.

Once the user has filed the document and the server has accepted the document, the server enters the page count and the IDs of the areas in which the files and the quicklook are saved, into the table. The data in the table `doclock` are deleted. This unlocks the document.

If the user does not file the document and cancels the procedure, the entries in the table `object#`, `sdrel` and `doclock` are deleted.

Both actions, the creation of the DMS object and the creation of the document files are logged in the table `osobjhist` as individual actions and are all listed individually in the clients' editing histories.

Scanner

The editing modules process data through the TWAIN or KOFAX interface of a scanner. If the scanner is installed according to the manufacturer's specifications and access was successfully tested, the editing modules can access the scanner. To test access through the KOFAX interface, KOFAX provides the test application

VCDEMO.EXE, access through the TWAIN interface can be tested using the Imaging application, which is part of the operating system.

When installing the KOFAX components, the path to the component directory `imgctls\bin` must be specified. This directory must be accessible by the editing module through the system variable PATH. If this is not the case, the path to the directory must be entered manually. To communicate with SCSI adapters, the editing modules require the latest ASPI drivers.

Scanner settings

Scanner settings are user-specific. Workstation-specific use of scanner settings can be enabled with settings in the client.

The data are managed in the database in the table `osconf`. The data are saved for each user (column `osconfuser`) under the configuration name `axscan` (column `osconfname`) in the column `osconfal`.

To access these data, the file `axscan.ini` with the contents of the database field is created and saved in the user-specific directory `etc\user\user_name`.

The file contains interface, workstation and editing module specific sections containing settings. The KOFAX section contains references to individual configurations which are also saved in the table `osconf`. If one of these configurations is needed, the data are also transferred into a file named `configuration_name.ini`, saved in a user-specific directory `etc\user\user_name` and used for the scanner settings.

Creating Windows Documents

Windows document are documents which are not created with editing modules of enaio® client, but with external applications which are launched by the client. Templates are set up for Windows documents which can be selected by the user and which launch the linked applications when creating a document.

Templates and applications for Windows document types are administered in the template administration of enaio® administrator. All templates are saved by enaio® server into directory `etc\Templates`.

In the database, data of the template administration are saved in a number of tables:

- templates and corresponding references to applications in the table `ostemplate`,
- templates and corresponding references to document types in the table `ostempobj`,
- templates and corresponding references to groups in the table `ostempuser`,
- application data in the table `osapplications`.

The table `ostemplate` has the following structure:

Column	Contents
`id`	the unique ID of the template
`app_id`	the unique ID of the application, the data of the application are listed in the table `osapplications`
`viewer_id`	the unique ID of the viewer, the data of the viewer are listed in the table `osapplications`
`namespace_id`	ID of the namespace, the namespace name is saved in the table `osnamespace`
`alias_id`	the ID of the alias, the alias name in saved in the table

	`osalias`
`flags`	flag, indicating if the template is located in the template directory
`name`	file name of the templates
`extension`	file extension of the template
`checkoutdir`	directory in which the editing template was saved
`checkoutstation`	workstation on which the template is being edited
`checkoutuser_id`	user which is editing the template

Table 15

The table `ostempobj` has the following structure:

Column	Contents
`temp_id`	the unique ID of the template
`object_id`	ID of the document type as main or sub type

Table 16

The table `ostempuser` has the following structure:

Column	**Contents**
`temp_id`	the unique ID of the template
`user_id`	the unique ID of the assigned user group or an assigned user
`flags`	flag, if the entry `user_id` indicates a group or a user.

Table 17

The table `osapplications` has the following structure:

Column	**Contents**
`id`	the unique ID of the application
`flags`	flag on how to launch the application
`name`	Name of the application
`path`	path with mapped drive name and file name of the application

Table 18

The client loads the data from the database. If data of the W template administration are changed in enaio® administrator, the client does not need to be restarted.

Example of template data in the database:

The table `ostemplate` contains template data and the application assignment:

id	**app_id**	**viewer_id**	**namespace_id**	**alias_id**	**flags**	**name**	**extension**	**checkout**
300	139	139	240	241	0	ef.doc	doc	

The table `osapplications` contains the application data for the application with the ID 239.

id	**flags**	**name**	**path**
139	2	MSWord	C:\Program files\Microsoft Office\Office\WINWORD.EXE

The table `osnamespace` contains the name of the namespace with the ID 240.

id	**name**
240	Office

The table `osalias` contains the name of the alias with the ID 241.

id	name
241	Standard letter

The table `ostempobj` contains the assignment of templates to document types.

temp_id	object_id
300	262284

The table `ostempuser` contains the assignment of templates to users.

temp_id	user_id	flags
300	625	1

Document Files of Windows-Documents

As opposed to image documents, templates are used for Windows documents and exactly one file in the application format is created in the archive for a Windows document. If a user creates a document, he indexes it first and the client transfers the data in the same way as image files into the respective tables. Then the user selects a template.

If the user does not select a template but cancels the process at this point, the indexing data will remain as a document without pages.

enaio® server transfers the template from the template directory `\templates` to the client which saves it in the CACHE area and sends it back to the server. The server saves the templates as a document in the WORK area.

Path and file extensions have the same structure as image documents, the main type of Windows documents is `4`, and the file extension corresponds to the extension of the template.

```
\WORK\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

Path and file name in the client have the same structure:

```
..\[user id]\CACHE\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

The client starts the application as defined in enaio® administrator during configuration of the application by executing the program via the specified path or via the application which is registered for the file extension.

If the user saves the document, it is saved in the CACHE area. If the user checks in the document, it is transferred to the server, saved by the server in the WORK area and made available for other users for editing. It is automatically deleted from the CACHE area.

Table Structure when creating a Windows Document

Similarly to the creation of image files, the client obtains an ID from the server of the document and enters the indexing in the table `object#`. The page count is '0', the user is entered in the column `lockuser`. Document data and user information are entered in the table `doclock` to indicate the checkout.

The document location is entered in the table sdrel.

If the user selects a template, the client requests the template from the server, saves it in the CACHE area and transfers it to the server which creates the document with the template from the template directory and enters the page number '1' and the media IDs in the table object#. The client starts the application with the document. If the user checks in the document, it is transferred to the server, saved by the server in the WORK area and deleted from the CACHE area.

The user is removed from the table `object#`, the entries in the table `doclock` are deleted. This unlocks the document. The creation of DMS objects and documents with a template on the server and opening for editing with an application are logged in the table `osobjhist` as individual actions and are all listed individually in the clients' editing histories.

Transfer of Windows Documents

Instead of choosing a template, the user can select a file from the file system and choose if the application will be opened. The user can only select files with a file extension which matches the template which is assigned to the document type.

If a user selects a file, the file is transferred by the client to the server. The server saves the file in the WORK area. If the user wants to open the document, it is transferred by the server to the client and it is checked out. The client opens the document with the application which corresponds to the file extension.

Transfer E-Mails

Users have the following options to transfer e-mails into the archive:

- from the mailbox
- by dragging and dropping from Microsoft Outlook
- using enaio® document-storage

The document type e-mail is set up, just as any other document type with enaio® editor. The data are saved in the tables `osobjdef` and `osobjfields`, the indexing is saved in the respective table `object#`.

enaio® supports e-mail transfer from the mailbox for all MAPI 1.0 compliant Windows mail systems and IMAP systems with SMTP/MAPI.

To enable e-mail server communication, with enaio® enterprise-manager mailboxes can be switched from MAPI to IMAP across the entire system. Data is expected in MIME format and also managed in this format in the archive.

The client determines with the user-specific settings if users use the mailbox. The settings are saved for each individual user in the table `osconf`. In this table user-specific settings which are loaded by the client at startup are stored for every user (column `osconfuser`) as type `3` (column `osconftype`) in the column `osconfval`. To initialize the inbox the section `[CLIENT]` contains the entry `MAPIINIT`. Value `1` initializes the inbox, a value of `0` does not.

The mailbox of the user who is logged on to the operating system will open.

The initialization of the inboxes can be dictated by an entry in the configuration file `as.cfg` for the entire system. The path to this file is saved in the table `osresources`. The file can be edited with any editor. The entry is part of the section `[SYSTEM]` and is also called `MAPIINIT`. The default value is '1'. Thus, the inboxes are enabled by default. In addition, the system must be licensed respectively, though.

E-mails contain header information, e.g. 'From', 'To', 'Subject'. The data can be assigned to index fields of the document type e-mail. The configuration file can also contain assignment with internal names.

Mailbox

Data mapping is performed automatically when filing e-mails from the mailbox with the help of internal names for the e-mail document type in enaio® editor.

The fields require the following labels:

Data	Internal field name
From	MAIL_FROM
To	MAIL_TO

CC	MAIL_CC
BCC	MAIL_BCC
Date	MAIL_SUBMIT_TIME
Subject	MAIL_SUBJECT
Message	MAIL_BODY

Table 19

This procedure is not case-sensitive and since internal names are used, the labels on the data form are disregarded. If no matching index fields are set up for these data, the data are not used for the indexing, but displayed in the e-mail module.

Drag & drop from Microsoft Outlook

The assignments for transferring e-mails with drag & drop from Microsoft Outlook are also specified with internal names. Due to compatibility reasons with older versions assignments can also be loaded from the configuration file `as.cfg`. If settings can be found there, the assignment by internal names will not be used. (When starting the client, the server creates a copy of the file `as.cfg` in the directory `\Temp\ostemp\'ServerName'#'Port'\etc\`.) If no assignments between e-mail and index fields are set up, the data will not be used for the indexing, but displayed in the e-mail module.

Document Files of E-mail Documents

As for other document types, the client transfers an e-mail into the WORK area and transfers it after indexing to the CACHE area.

Path and file name have the following structure:

```
..\[user id]\CACHE\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

File extensions of e-mails are `mim`, `ima` or `msg`.

Then the client transfers the file to the server.

Path and file name on the server have the following structure:

```
\WORK\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

Once transferred, the file will be deleted from the WORK area and the cache.

Table structure when storing E-mails

Similarly to other document types, the client obtains after automatic or manual indexing of the e-mail an ID from the server of the document and enters the indexing in the table `object#`. The document location is entered in the table `sdrel`. Then the document is saved in the WORK area and deleted from the CACHE area.

Both actions, the creation of the DMS object in the database and the saving of the document files are logged in the table `osobjhist` as individual actions and are all listed individually in the clients' editing histories.

enaio® document-storage

enaio® document-storage provides features for archiving e-mails directly out of Outlook, Lotus Notes or GroupWise in enaio®.

These features are accessible through a toolbar in the e-mail application. A clear user interface is available for configuration purposes.

Furthermore, enaio® document-storage allows you to automatically flag e-mails in various ways, to include or exclude attachments from archiving and much more.

The system role 'Editor: Start' is required for the configuration of enaio® document-storage.

The configuration management is where you define to which enaio® object types e-mails are assigned, and which characteristics are saved as index data. As well, the attachment proceeding is set. E.g. attachments can be managed as individual documents. In such a case, it can be checked whether an attachment has already been archived in enaio®, and, instead of archiving the attachment again, you can decide to have a reference created.

What is more, based on imported e-mails you can run workflow processes. As process variables you can pass the e-mails' characteristics.

The configuration file `axvbdocstorage.xml` is saved to the `\etc` directory in the data directory. If environments include several server groups and thus have more than one data directory, just copy the configuration file to all etc directories available.

Function Overview

Feature	Microsoft Outlook (2007 or greater)	Novell Group Wise (v7.01 or greater)	IBM Lotus Notes (v6.5.6 or greater)
Allow separate filing of attachments	✓	✓	✓
Allow filing e-mails without attachments	✓	✓	✓
Move to archiving folder	✓		
Archive and delete object	✓		
Remove attachments	✓		✓
Insert OS link file	✓		✓
Insert archiving ID in body	✓		✓
Archive, send and determine Outlook location	✓		
Auto file at identified location	✓		
File object in MSG format	✓		
Save signed e-mails in MSG format	✓		
Copy e-mail properties	✓	✓	
Update object	✓		
Use enaio client for archiving	✓	✓	✓
Transfer individually	✓	✓	✓
Address resolution using LDAP	✓		✓
Use cooperation mode	✓		

Create references	✓		
Connect attachments using notes	✓	✓	✓
Deduplication	✓	✓	✓

Transfer Video Documents

The module for videos can be used to transfer videos in MPEG and AVI format from the file system. These will be administered in the original format in the archive, but the extension will be changed to `ASM` and `000` in the CACHE and WORK area. You can only assign one file to a video document. If slides are created for the video document, the first frame is extracted and saved in TIF format with the extension `DIA`. File management and table structure correspond to those of image documents. The main type of video document types is `5`.

Transfer XML Documents

XML documents are either imported or transferred from the file system into the client via a file selection dialog. You can only assign one file to an XML document. The files are not displayed before being filed. To view XML document types, stale sheets must be assigned.

If a user creates an XML document, he indexes it first and the client transfers the data into the table `object#` and the location data into the table `sdrel`. Then the user selects an XML, XSL or XSLT file in the file selection dialog. If the user cancels this procedure, the data will be deleted. The selected file is copied into the WORK area, from there to the CACHE area and to the server. The server saves the file in the WORK area. File management and table structure correspond to those of other documents. The main type of XML document types is `7`.

Transfer from the File System

Windows documents, image documents and video documents can be directly dragged by the user from the file system, e.g. Windows Explorer into an open folder. The client copies the file into the workspace and creates a list of document types based on the file extension to which the file could be assigned. The assignment data for Windows document types are derived from the table `ostemplate`.

If the extension cannot be associated with any editing module or Windows application, the user will receive an error message.

If the user indexes the document, the client enters the data into the table `objekt#`, the location into the table `sdrel`, copies the file into the cache and transfers it to the server.

Path and file name on the server have the following structure:

```
\WORK\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

Path and file name in the client have the following structure:

```
..\[user id]\CACHE\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

The file will not be opened. The creation is logged in the table `osobjhist`.

Transfer with enaio® printer

During the installation of enaio® two printer drivers can be installed which can be used by the user in any application with printing ability to transfer files as image documents or as PDF documents into the DMS. PDF documents are TIFFs integrated into the PDF format.

The enaio® printer(BW) prints in black/white in the format TIF G4 or PDF, the enaio® printer(Color) prints in color in the format JPEG or PDF. Black/white images are assigned to a black/white document type, color images to a color image document type and PDF documents to a Windows-document type.

If no PDF application is assigned to a Windows document type, PDF documents can still be assigned. The user will then receive a notification. PDF documents can then only be opened with a registered application as read-only.

The black/white printer can also use background images. These background images must be available as black and white bitmaps in TIFF G4 format and have the same size as the document pages. Format and background images are managed in enaio® administrator. If the PDF format is selected, users can enable a confirmation dialog in their user settings which allows them to choose a different format than PDF:

enaio® administrator saves the settings in the `as.cfg` configuration file. The path to this file is saved in the table `osresources`.

The entries in the file have the following structure:

```
[ASPRINT]
HINTERGRUND0101=1.Brief,C:\HG_Bilder\1brief.tif
HINTERGRUND0102=ff Brief,C:\HG_Bilder\2brief.tif
HINTERGRUND0201=1.Auftrag,C:\HG_Bilder\1auftrag.tif
HINTERGRUND0202=ff Auftrag,C:\HG_Bilder\2auftrag.tif
DDOCFORMAT=0
PDOCFORMAT=1
```

The section `[ASPRINT]` contains a list of background images. Images with `1` as the last digit are offered as the first page, images with a `2` as the last digit for the following pages. The path information is saved along with the drive information. All users must be able to access images this way.

The value of `DDOCFORMAT` dictates the format for the black/white printer, `0` equals TIF, `1` equals PDF. The value of `PDOCFORMAT` dictates the format for the color printer, `0` equals JPEG, `1` equals PDF.

The printer drivers load the data from the file `as.cfg`, which is written at client start into the directory `\Temp\ostemp\'ServerName'#'Port'\etc\`. The path to this file can be obtained from the client. If the client is not launched, the user will receive an error message.

The printer drivers provide pat and file name of the files to the client. The user selects the location in the DMS, an open folder or the filing tray, specifies the document type and indexes the document. The client enters the indexing and location data in the database and transfers the document files to the server. File management and table structure correspond to those of other documents.

enaio® office-utilities

With enaio® office-utilities users can also access printer drivers and choose a format. If background images are set up, they can also be selected. The enaio® office-utilities create the configuration file `asprint.ini` in the directory `WINNT`. This file contains the format selected by the user. The printer drivers will first look for a configuration file at startup and apply the format settings from this file.

The location and the indexing are then entered in the same ways as for the direct access of the printer driver.

Module Spanning Documents

Document types can receive the property 'module-spanning' in enaio® editor. For these documents the user selects an editing module after indexing. This property is managed in the column `flags` of the first dialog element of the DMS object in the table `osobjfields`. Module-spanning document types have a main type depending on which main type was selected in enaio® editor at creation. This main type is entered in the column `haupttyp` of the table `object#`. The creation and the table structures are the same as for other document types.

Documents without Pages

Documents without pages are documents of a document type which consist only of the indexing with no assigned files. Document types can receive the property 'w/o pages' in enaio® editor. In this case no user can add files to it. This property is managed in the column `flags` of the first dialog element of the DMS object in the table `osobjfields`. Users can create documents without pages if they only want to save indexing data or want to assign files at a later time. Then in the column `anzahl` of the table `object#` the value '0' is entered and no editing module is opened.

The Filing Tray

Users can either transfer documents into an open folder or into a register, or file them in the filing tray. The filing tray is a user-specific area which can only be accessed by the user. Objects in the filing tray are not included in queries. The menu item 'VIEW USER TRAY' in enaio® administrator displays a list of contents for each user. If a user is deleted in enaio® administrator, the contents of the filing tray can be moved to the user who deletes the user. Documents created by the user for the filing tray are assigned to a document type or filed without type.

Document with Type

Documents with a document type are indexed, the indexing is entered in the table `object#`. The location is not entered in the table `sdrel`, but in the table `mdrel`. This table contains all documents from filing trays and all documents which are located in portfolios and has the following structure:

Column	Contents
`loeschen`	currently not used
`zeitstempel`	the exact time when a DMS object was filed
	the unique ID of the portfolio/filing tray
`mappe_id`	the unique ID of the document
`object_id`	Folder ID (portfolio)
`stamm_id`	ID of the document type
`objekttyp`	main type of documents without type
`modul`	currently not used
`eingang`	the creator of documents without type
`absender`	currently not used
`ausgang`	number of pages of documents without type
`anzahl`	

Table 20

Filing trays and portfolios are managed in the table `mappe`:

Column	Contents
`id`	the unique ID of the portfolio/filing tray
`delete`	currently not used
`time stamp`	the exact time when the portfolios were created
`created`	the creation date of portfolios
`creator`	the creator of portfolios
`recipient`	the recipient of portfolios or the owner of the filing tray
`topic`	the topic of portfolios
`type`	flag, if a portfolio is public
`links`	flag for portfolios, if notes exist
`creator_id`	currently not used
`recipient_id`	currently not used

Table 21

The file management of documents from the filing tray corresponds to the one of documents from folders. Path and file name on the server have the following structure:

```
\WORK\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

Path and file name in the client have the following structure:

```
..\[user id]\CACHE\[main type]\[sub type]\[mod 0x100]\[DocID].[file extension]
```

Documents without type

Documents which are filed by a user in the filing tray without a type are not indexed. The user selects an editing module and selects files. The client saves the files in the WORK area. If the user files a document, the client saves the files in the cache and transfers them to the server.

Windows documents without type cannot be created in the filing tray.

Path and file name on the server have the following structure:

```
\WORK\[main type]\[DocID].[file extension]
```

Path and file name in the client have the following structure:

```
..\[user id]\CACHE\[main type]\[DocID].[file extension]
```

Documents without type are entered in the table `mdrel` in the database.

The creation of documents in the filing tray is logged in the table `osobjhist`.

If you want to move documents without type from the filing tray into the system, they must be assigned to a document type and indexed first.

The filing tray behaves according to the rights system. If a user creates a document which he is not permitted to delete due to restrictions of his access rights, he can also not delete it from his filing tray.

Working with Documents

General Requirements

In order to edit documents, enaio® client requires the current object definition, object relations and configuration entries which it loads at startup. It also requires the current rights of the logged-on user. The data can be updated in enaio® client with the shortcut Shift + Ctrl + F5.

DMS objects are integrated into a hierarchical structure and are indexed. User will require basic knowledge of the hierarchical structure and the object-specific indexing structure in order to search for DMS objects.

Users search for DMS objects

- by entering search criteria for the indexing and the basic parameters and receiving hit lists as a result
- by following the hierarchical structure, i.e. opening folders and registers and viewing tables of contents
- by using the full text search which is based on the indexing and the document content and receiving hit lists as a result
- by executing saved queries and receiving hit lists as a result

When executing a search, enaio® client creates SQL statements which are based on the user input and the data derived from the rights system. The statements are then sent to the enaio® server that parses them and queries the database.

The general parser `oxtrodbc.dll` is used as a parser. For the Oracle database the parser `oxoratnl.dll` can be used, for Microsoft SQL Server the parser `oxmsstnl.dll`.

The full text index server is accessed for a full text search.

The client receives a list of DMS objects which it displays as a hit list or in a folder window as a content list. Users can configure search structures, save and provide them to other users. The display of DMS objects in hit lists and folder windows can be configured. DMS objects can be combined for quicker access to personal and public areas.

enaio® client manages user-specific settings in the table `osconf`.

The table has the following structure:

Column	Contents
`osconfuser`	the unique user ID
`osconfstate`	currently not used
`osconftype`	the configuration type
`osconfval`	the configuration data
`osconfid`	the unique ID of the configuration
`osconflink`	the unique user ID of the assigned profile
`osconfname`	file name, if the configuration data is provided to

	components as a file

Table 22

The following configuration types are used for user settings:

osconftype	Contents
1	saved queries
2	Links to external programs
3	presets, window positions, cached data
	Folder structure in the object search
4	SQL Queries
5	DMS objects in the object search
7	

Table 23

If a profile is assigned to a user, the configuration type data stored in the profile are also transferred and entered in the database.

If a user does not have the system role 'Save own settings', he can temporarily change the data of these configuration types in the client, the changes will not be saved in the database, though.

Not user-specific settings are managed in the file `as.cfg`. The path to this file is saved in the table `osresources`. When starting the client, the server creates a copy of the file `as.cfg` in the directory

`\Temp\ostemp\'ServerName'#'Port'\etc\.`

The client loads the settings from this file.

Window positions are loaded by client components from the user settings in `osconf`. Entries of the configuration type 3 correspond to these user settings.

If user settings are saved with errors in the system, users may press and hold the F8 key during startup. The client will then not load the window positions but initialize with default settings.

In the following, only a few aspects regarding the understanding of the system are explained. More detailed instructions on working with documents can be found in the enaio® client handbook.

Search

When executing a search, enaio® client queries the database with SQL statements which are based on the user input and the data derived from the rights system. The full text server is accessed for a full text search.

The search is restricted to DMS objects which are assigned to a folder type within the hierarchical structure. Overall searched are possible with extended queries only.

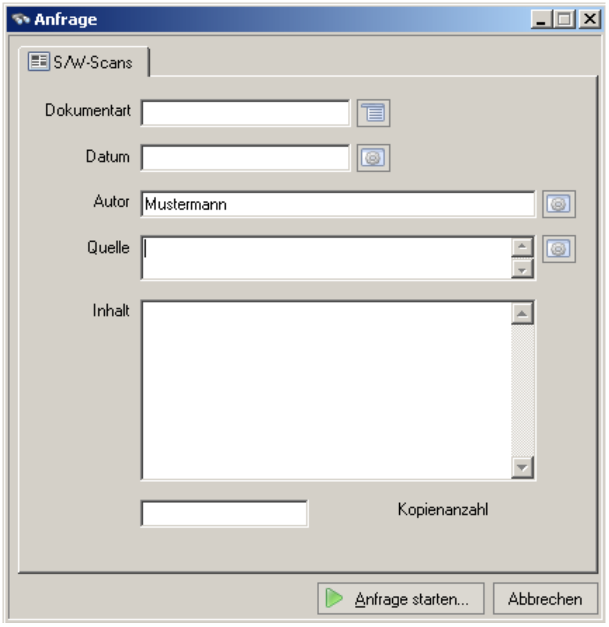
The user can specify for a query if a wildcard is inserted for any number of characters before or after the search criteria. This user-specific setting is saved in the table `osconf` along with configuration type `3` data. The option is labeled `autoasterisk='value'`

Search criteria in a search form are linked with the logical AND.

The indexing of the dialog element TABLE is not available for the search.

An example search:

The user opens a document search form and enters search criteria.



Dokumentart
Datum
Autor Mustermann
Quelle
Inhalt
Kopienanzahl

Figure 21 Example of a search form

As a default setting, a wildcard for any number of characters is inserted before and after the entries.

enaio® client forms the following statement:

```
select     o.feld1,o.feld3,o.feld4,o.id,o.links     from     objekt2     o
where ( {fn UCASE(o.feld1)} like '%Mustermann%' and
({fn UCASE(o.feld6)} < '300'));
```

The indexing of the fields which will then be displayed in the hit list is queried. The query criteria is the entry in the field `Autor` (document field 1). The second query criterion for the field `Kopienanzahl` (document field 6) results due to a logical expression of the rights system. The user can in this case only view document indexing for which the copy number is smaller than `300`.

Hit lists

If hit lists contain one document only, it can be instantly opened. This user-specific setting is saved in the table `osconf` along with configuration type `3` data. All hits of a full text query – folders, registers and documents – are listed in a hit list. It is not distinguished if the search term was part of the indexing or the content.

The client determines if documents are being edited by other users and are therefore locked via the indexing table `object`*`n`*. In such case these documents are flagged in the hit list.

Folder Window

To open folders and registers, enaio® client creates SQL statements which query all DMS objects with a location assigned to the selected folder or register. Objet relations are in this case not relevant. The client determines if documents are being edited by other users and are therefore locked via the indexing table `object#`. In such case these documents are flagged in the hit list.

Filters are saved with user-specific settings for the display of folder windows in the table `osconf` with configuration type `3` data and can be assigned to users via the profile administration.

The filter entry has the following structure:

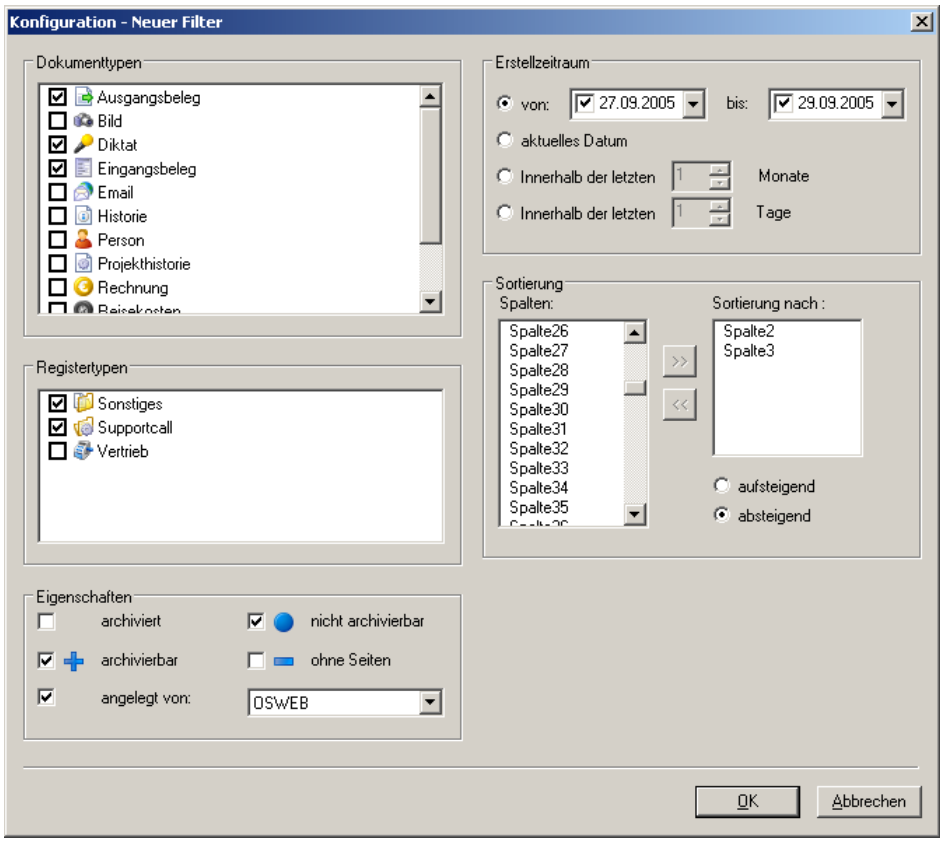
```
[FILTER@2]
Configs=Rechnungen,Belege
[FILTER@2|Rechnungen]
...
```

For each folder type the section [FILTER@'sub type'] is created in which the set up filters are listed.

For each filter the section [FILTER@'sub type'|'filter name'] is created in which the set up filter criteria are listed.

An example filter

Configuring a filter in enaio® client:


Konfiguration - Neuer Filter

Dokumenttypen
- [✓] Ausgangsbeleg
- [ ] Bild
- [✓] Diktat
- [✓] Eingangsbeleg
- [ ] Email
- [ ] Historie
- [ ] Person
- [ ] Projekthistorie
- [ ] Rechnung
- [ ] Reisekosten

Registertypen
- [✓] Sonstiges
- [✓] Supportcall
- [ ] Vertrieb

Eigenschaften
- [ ] archiviert
- [✓] nicht archivierbar
- [✓] archivierbar
- [ ] ohne Seiten
- [✓] angelegt von: OSWEB

Erstellzeitraum
- [✓ von:] [✓] 27.09.2005 bis: [✓] 29.09.2005
- aktuelles Datum
- Innerhalb der letzten 1 Monate
- Innerhalb der letzten 1 Tage

Sortierung
Spalten: Spalte26, Spalte27, Spalte28, Spalte29, Spalte30, Spalte31, Spalte32, Spalte33, Spalte34, Spalte35
Sortierung nach : Spalte2, Spalte3
- aufsteigend
- [✓ absteigend]

Figure 22 Configuring filter criteria

Entries in the database:

```
[FILTER@1|W-Doks]
TypesToShow=262298|262284|262317|
CreatedByUser=OSWEB
SortColumns=2|3|
CHECK1=0
CHECK2=1
CHECK3=1
CHECK4=0
Active=0
TimeFrom=4294967295
TimeTo=4294967295
UseActDate=0
UseFromTo=0
UseLastMonths=1
UseLastDays=0
LastDaysMonths=1
ShowRegister=0
```

```
SortAscending=0
```

Saved Queries

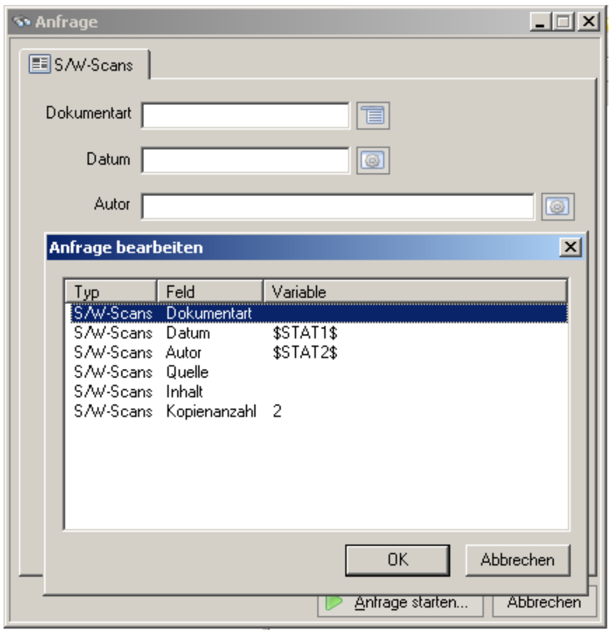
Queries which are created by users with simple query forms and query forms in expert mode can be saved. Users can add variables to fields for saved queries. If saved queries are executed, only fields with variables are displayed.

Entries for a full text search are not saved.

Saved queries are saved by enaio® client in the table `osconf` under the configuration type `1` and displayed in the object search. They can be assigned to users via the profile administration.

An example of a saved query:

A folder query is saved with a static variable. Static variables are variables for which the last entry is saved temporarily. It is used as a preset value at next execution, but not saved when the client is exit.



Typ	Feld	Variable
S/W-Scans	Dokumentart	
S/W-Scans	Datum	$STAT1$
S/W-Scans	Autor	$STAT2$
S/W-Scans	Quelle	
S/W-Scans	Inhalt	
S/W-Scans	Kopienanzahl	2

Figure 23

The query is saved under the query name 'SW Scan' and receives the property 'Default action: open'.

The entry in the table `osconf` looks as follows:

```
[1@0]
#OSPOS000#=$STAT1$
#OSACT#=1
#OSPOS005#=1*
[SYSTEM]
NAME=S/W-Scans Kopienanzahl 2
IDENT=1509
VARREQUEST=1
DEFACTION=0
```

The section [Objekttyp@0] indicates the saves query for the object type. These entries are assigned to the fields via the tab position (`#OSPOS000#`).

The entries are saved according to the tab position of the fields. If the tab order is changed, saved queries are executed incorrectly.

SQL Queries

SQL queries are queries which allow users to access data in the database and execute VB scripts, regardless of the access rights to DMS objects and regardless of the hierarchical structure.

SQL queries cannot be used to delete or manipulate data from the database with SQL commands or VB scripts.

Extended queries can only be set up by users who have the 'Editor: Start' and the 'Client: Edit SQL queries' system role.

SQL queries are saved by enaio® client in the table `osconf` under the configuration type `5` and displayed in the object search. They can be assigned to users via the profile administration or simply sent.

An example of an extended query:

Each index entry is queried, sex (feld9) and age (feld10) of a folder type (stamm2) for an anonymized statistical analysis. The columns are labeled in the results list (HEADER).

The user can run a VB script (SCRIPT1) which creates a Microsoft Excel table including these data by clicking a button (Export). Access permissions to the folder type are not required.

```
[P-Statistik]
IDENT=1250
SQL=select feld9,feld10 from stamm2
HEADER=Geschlecht,80;Alter,40
TREEPARENT=0
FLAG1=1
TEXT1=Export
FLAG2=0
TEXT2=
FLAG3=0
FLAG4=0
FLAG5=0
CHANGED=0
SCRIPT1=sub ExportToExcel()....
SCRIPT2=
SCRIPT3=
```

Displaying XML Documents

XML documents are displayed in a browser using corresponding style sheets. The assignment can be managed with the configuration file `as.cfg`. It contains a section for XML documents.

Example:

```
[XML]
objekt(458752)=Tabelle,Grafik
Tabelle=tabelle.xsl;html
Grafik=grafik.xsl;svg
default(458752)=Grafik
```

When opening an XML document, the default style sheet is used. More style sheets can be selected from a list. Style sheets must be saved to the directory `etc\Templates`. Users also have the option to select from various style sheets via a file selection dialog. If no style sheets are set up, the XML files are displayed.

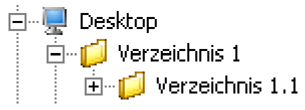
Folders in the Object Search

In order to manage references to DMS objects, links to external programs, saved queries, SQL queries, and search forms concisely in the object search, users can create a folder below the default folder 'Desktop'.

The data are saved in the table `osconf` with the configuration type `4`.

Example:

Folders in the client's object search:



Entries in the database:

```
[26279936]
#OSPOS000#=26279937
#OSPOS001#=26279939
[26279937]
NAME=Verzeichnis 1
#OSPOS000#=26279938
[26279938]
NAME=Verzeichnis 1.1
[26279939]
NAME=Verzeichnis 2
```

The folder 'Desktop' will be created automatically.

Data managed in the object search are saved including the folder number.

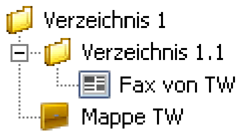
References to DMS Objects

Users can place references to DMS objects inside the object search in the folder 'Desktop' or in a newly-created folder and additionally manage them on the navigation.

The data are saved in the table `osconf` with the configuration type `7`. The unique ID, the object type and the folder assignment are stored (`TREEPARENT`).

Example:

Object search:



Entries in the database:

```
[Fax von TW]
#OSIDENT#=1436
#OSTYPE#=131099
TREEPARENT=26279938
POS=0
SYSMODE=0
[Mappe TW]
#OSIDENT#=1524
#OSTYPE#=13303808
TREEPARENT=26279937
POS=1
SYSMODE=0
```

When deleting documents it is not checked if references were created pointing to these objects.

Links to External Programs

Users can place references to external programs inside the object search in the folder 'Desktop' or in a newly-created folder and additionally manage them on the navigation area. These references can also be created via the 'SEND TO' menu. An external program can be just launched or launched with specific data.

The following parameters are used for data transfer:

- `%o`

  One or more paths to DMS-files are transferred directly. The space character is used as a separator. If the path or file name contains spaces, the file path is specified in quotation marks.
- `%p`

  A path to a text file which contains a path to a DMS file per line.
- `%i`

  A path to a text file which contains a unique ID and the object type – separated with a comma – per line. This parameter is useful if many objects are transferred.
- `%f`

  One or more paths to files in the archive. If the files are not yet in the CACHE area of the client, they will be transferred. Spaces are treated in the same way as for parameter `%o`.
- `%g`

  A path to a text file which contains a path to a file in the CACHE area per line. This parameter is useful if many objects are transferred. If the files are not yet in the CACHE area of the client, they will be transferred.

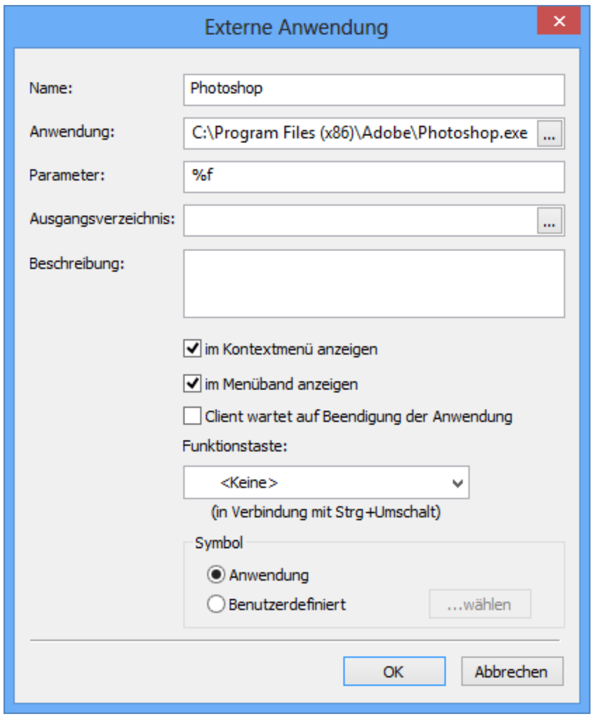
The function keys can be used together with the key combination Ctrl+Shift to launch external programs directly. If an icon is defined in such application it can be used in the tree view for the external program. However, it is also possible to assign a different icon to the program.

The transmitted data are not deleted by enaio® client. Files which were transferred into the CACHE area of the client are not flagged as checked out. These are automatically deleted from the CACHE area when exiting the client.

The configuration data are saved in the table `osconf` with the configuration type `2`.

Example:

Configuration: Entries in the database:


Externe Anwendung

Name: Photoshop

Anwendung: C:\Program Files (x86)\Adobe\Photoshop.exe

Parameter: %f

Ausgangsverzeichnis:

Beschreibung:

[✓] im Kontextmenü anzeigen

[✓] im Menüband anzeigen

[ ] Client wartet auf Beendigung der Anwendung

Funktionstaste: <Keine>

(in Verbindung mit Strg+Umschalt)

Symbol: [● Anwendung] ○ Benutzerdefiniert

```
[Photoshop]
EXECUTE=C:\Programme\Adobe\Photoshp.ex
e
PARAM=%f
SEND2MENU=1
ADDTOOLBAR=1
TREEPARENT=26279936
POS=6
```

Portfolios

Users can combine DMS objects in portfolios. Public portfolios are accessible to all users, not public ones only to the addressees specified by the creator and users to whom the portfolio was sent.

If a user opens a portfolio, only some access permissions apply:

- Logical expressions of access rights for the display inside the portfolio do not apply.
- DMS objects to which the user has no access because he has no access to the folder in which the objects are located, are displayed and can be edited.

DMS objects can be provided via portfolios to users which are not permitted to execute searches.

Filing trays and portfolios are managed in the table `mappe`:

Column	Contents
`id`	the unique ID of the portfolio/filing tray
`delete`	currently not used
`time stamp`	the exact time when the portfolios were created
`created`	the creation date of portfolios
`creator`	the creator of portfolios
`recipient`	the recipient of portfolios or the owner of the filing tray
`topic`	the topic of portfolios
`type`	public portfolio (`0`) or not public portfolio (`1`)
`links`	flag, if notes exist (`1`) or not (`0`)
`creator_id`	currently not used
`recipient_id`	currently not used

Table 24

If a user creates a DMS object inside a portfolio, enaio® client enters the data into the table `mdrel`. This table also contains all documents from filing trays and has the following structure:

Column	Contents
`loeschen`	currently not used
`zeitstempel`	the exact time when a DMS object was filed
`mappe_id`	the unique ID of the portfolio/filing tray
`object_id`	the unique ID of the DMS object
`stamm_id`	Folder ID
`objekttyp`	object type ID
`modul`	main type of documents without type (filing tray)
`eingang`	currently not used
`absender`	the creator of documents without type (filing tray)
`ausgang`	currently not used
`anzahl`	number of pages of documents without type (filing tray)

Table 25

Public portfolios can be edited by all users, not public ones only by their creator. The right to edit includes the right to delete a portfolio, remove DMS objects from the portfolio and to send the portfolio. Users who were sent a non-public portfolio can remove DMS objects from the portfolio, but not delete the portfolio itself. The data sheet of a portfolio can only be edited by the creator.

If a user deletes data from a portfolio, the entries are deleted from the table `mdrel`. If a user deletes portfolios, the data are deleted from the table `mappe` and also all entries in the table `mdrel` which refer to the portfolio.

Follow-ups

Users can set up a follow-up for documents, portfolios, registers, and folders. Follow-ups are set up for users or user groups. Addressees find follow-ups in the subscription area on the '**Follow-up**' tab if they have the necessary access rights.

Follow-up data are saved by the client in the table `osrevisit`. enaio® server informs all users for which the object is set as a follow-up.

The table `osrevisit` has the following structure:

Column	Contents
`id`	the unique ID of the DMS object
`objekttyp`	the object type
`user_id`	ID of the addressee
`firstvisit`	time of the follow-up
`lastvisit`	time when the DMS object's status is changed to 'edited'
`infotext`	annotations of the creator
`set_user_id`	ID of the creator
`set_time`	Date of creation
`osinformed`	time when the follow-up was displayed to a user on the tab
`ostype`	subscription mark
`osabotype`	action flag for subscriptions

Table 26

User with the system role 'Configuration security system' can view follow-ups of all users in the subscription area on the '**Follow-up**' tab. If a user marks a follow-up as '**edited**', the client enters the current date into the `lastvisit` column. Only if there is an entry, the follow-up can be deleted.

Subscriptions

User can subscribe documents, registers, and folders. Subscribed objects are listed in the subscription area as soon as data were changed. It can be selected if the DMS object is listed in the subscription area once the indexing or the document file was changed. Subscription data are saved by the client in the table `osabonnement`.

The table `osabonnement` has the following structure:

Column	Contents
`id`	unique ID of the subscription
`object_id`	the unique ID of the DMS object
`actionflag`	changes of the indexing (3), change of the file (4)
	currently not used
`produkt`	User name
`benutzer`	currently not used
`station`	flag if an e-mail is supposed to be sent
`channel`	specified e-mail address
`mail`	Info Text
`alias`	object type ID
`osobjtype`	

Table 27

If subscribed objects are modified, enaio® server writes data of the changed DMS object into the table `osrevisit`. It enters `1` into the table `ostype`, the action flag `3` or `4` into the column `osabotype`. It informs all of the users who have subscribed to the object.

If a user opens the '**Subscription**' tab, the client lists all DMS objects from the table `osrevisit` which are assigned to the user and which are flagged in the column `ostype` with `'1'`. Unedited subscriptions are indicated in bold letters.

If a user deletes an object on the '**Subscription**' tab, the entry in the table `osrevisit` is deleted, but the subscription itself is not removed. If a user deletes a subscription on the '**Subscribed objects**' tab, the object is deleted from the table `osabonnement`, but it is not verified if this object is listed in the table `osrevisit`.

Notes and References

Users can manage text notes and references from DMS objects, documents, registers, folders and portfolios to other DMS objects in a notes window. Text notes are created with a notes editor. References are created by dragging a DMS object with the mouse into the notes window. For this DMS object a reference is automatically created to the corresponding DMS object in the notes window.

If a user inserts text notes or references into the notes window of a DMS object, the client writes the number into the `links` column of the indexing table of the DMS object.

References are entered into the table `hyplnk`.

The table has the following structure:

Column	Contents
obj_id1	the unique ID of the DMS object
obj_typ1	the object type of the DMS object
obj_id2	the unique ID of the corresponding DMS object
obj_typ2	the object type of the DMS object
medium	currently not used
name	currently not used

Table 28

Text notes are entered into the table `remarks`.
The table has the following structure:

Column	Contents
id	unique ID of the note
parent	the unique ID of the corresponding DMS object
parenttype	the object type of the DMS object
medium	the ID of the media on which the note file is saved
type	the color of the note
stamp1	Time of creation
user1	Creator
stamp2	Time of last editing
user2	Editor

Table 29

Label and path for the note file are created with the ID and the object type of the corresponding DMS object.

Path and file name in the WORK area in the client have the following structure:

```
..\[user id]\REM\[main type]\[sub type]\[2NoteID]\[NoteID].txt
```

`[benutzerid]` is a user-specific directory which is labeled with the hex ID of the user.

`[Haupttyp]` is the hexadecimal main type of the corresponding DMS object.

`[Untertyp]` is the hexadecimal sub type of the corresponding DMS object.

`[mod 0x100]` corresponds to the last two digits of the hexadecimal ID of the note,

`[NotizID]` is the hexadecimal ID the note.

Path and file name on the server have the following structure:

```
\NOTE\[Haupttyp]\[Untertyp]\[2NotizID]\[NotizID].txt
```

The path to the note directory is saved in the table `path`.

If a user deletes a text note, the file is deleted from the work area and on the server. The data from the table `remarks` are deleted, the data in the indexing table are changed. If a user deletes a reference from the notes window, the data from the table `hyplnk` are deleted and the data in the indexing tables of both objects are changed. All users with access rights to DMS objects can open the notes window and edit or delete text notes or references. The creation and removal of references is logged in the table `osobjhist` and displayed in the editing history.

These actions are also entered as modification data in the indexing tables and displayed in the object information of the documents in addition to the basic parameters.

Boilerplate Text

Every user can save up to six files as boilerplate texts in the notes editor.

Path and file name of the boilerplate texts are saved in the table `osconf` with configuration type `3` data.

The entry has the following structure:

```
[NACHRICHT]
TEXT1=C:\Daten\Textbausteine\1textbaustein.txt
TEXT2=C:\Daten\Textbausteine\2textbaustein.txt
...
```

If a profile is assigned to a user, the boilerplate text entries stored in the profile are also transferred.

Exporting Data

Users can export the indexing of DMS objects and document files. The indexing of DMS object of the same type can be copied into the clipboard and inserted into a different application from the clipboard. User can also create configuration and export the indexing into a text file or an XML file.

Document files are exported in the stored format. In case of image documents individual pages or all pages can be exported in different export formats which can be specified.

Export Indexing

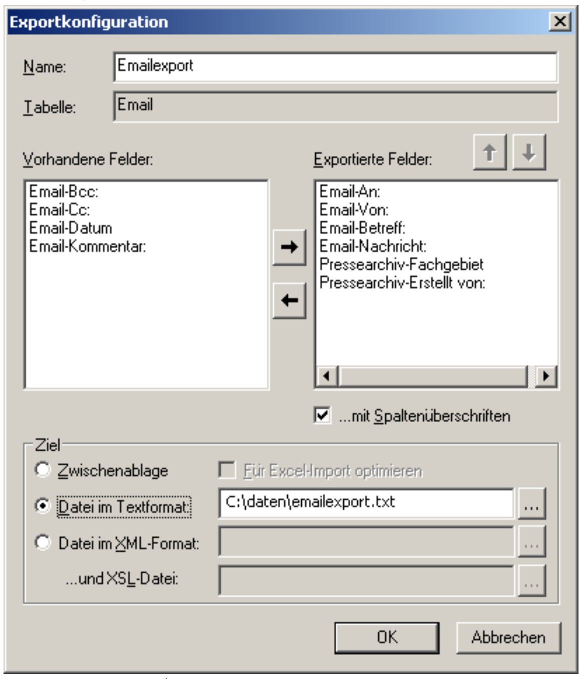
To export the indexing the user chooses for the indexing for the selected objects of the same type displayed in the hit list or a folder window if it will be exported or if a saved configuration will be used. A configuration stores all fields of an object type for which the indexing will be exported. For registers folder fields can be specified in addition, for documents the register and folder fields.

The indexing can be copied into the clipboard or exported into a file.

The configuration data are saved in the table `osconf` with the configuration type `3`.

Example:

Configuration:


Exportkonfiguration

Name: Emailexport

Tabelle: Email

Vorhandene Felder:
Email-Bcc:
Email-Cc:
Email-Datum
Email-Kommentar:

Exportierte Felder:
Email-An:
Email-Von:
Email-Betreff:
Email-Nachricht:
Pressearchiv-Fachgebiet
Pressearchiv-Erstellt von:

[✓] ...mit Spaltenüberschriften

Ziel
○ Zwischenablage
☐ Für Excel-Import optimieren
[● Datei im Textformat:] C:\daten\emailexport.txt
○ Datei im XML-Format:
...und XSL-Datei:

Entries in the database:

```
[ZWAL#Emailexport]
FLAGS=3
FILE=C:\daten\emailexport.txt
ORDTYPE=1
DOCTYPE=393218
FIELD00=o.feld11
FIELD01=o.feld4
FIELD02=o.feld3
FIELD03=d.datum1
FIELD04=d.feld1
FIELD05=d.feld2
FIELD06=d.feld4
```

Data records are separated by page break characters, individual data in each row by tabs.

Exported XML files have the following structure:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <!-- optimal systems GmbH -->
- <Exportliste>
- <Zeile>
 <HFeld>SpaltenüberschriftFeld1</HFeld>
 <HFeld>SpaltenüberschriftFeld2</HFeld>
 </Zeile>
- <Zeile>
 <DFeld>VerschlagwortungFeld1</DFeld>
 <DFeld>VerschlagwortungFeld2</DFeld>
 </Zeile>
- <DFeld>VerschlagwortungFeld1</DFeld>
 <DFeld>VerschlagwortungFeld2</DFeld>
</Zeile>
 </Exportliste>
```

If users specify an XSL file, the XML file is formatted with the XSL file and transformed into an HTML file.

Export Files

If files are exported from hit lists or folder windows, the files are saved in module-specific file formats, Windows document files in the application format.

Black and white images are saved in the module-specific file format TIFF G4, multi-page documents as Multipage TIFF.

The name is automatically created according to this schema:

```
DOC[number][page number].[file extension]
```

`[number]` is a three digit number of the exported files.

`[page number]` is the hexadecimal three digit page number. Black and white images and video documents are always labeled with '000'. Windows documents which always consist of one file do not have this part.

During export it is not verified if documents with the same name exist in the destination directory. If this is the case, all files are overwritten. The exported files contain annotations on transparencies which cannot be edited by the user. Annotations which can be edited by the user are not exported.

For black/white, grayscale and color images individual pages can be exported in a selectable format with the display module. Use the 'SAVE PAGE AS' function from the 'FILE' menu to save the displayed page under the specified name and format.

Users can choose the 'EXPORT PAGES' function to open the export dialog and specify in detail which pages will be exported, using with name schema, in which format and if annotations which the user can edit will be burnt in just as the annotations he cannot edit.

The export of files is logged in the table `osobjhist` and displayed in the editing history with the information 'OBJECT OUTPUT AS READ-ONLY'.

Sending DMS Objects

When sending DMS objects via the context menu 'SEND TO...' within hit lists, enaio® client launches the installed (MAPI) e-mail client and attaches the files to the e-mail message. If Microsoft Outlook is installed, users can also directly choose

Microsoft Outlook for sending. In this case the Outlook Add-In including the 'SEND AND ARCHIVE' feature becomes available.

User can send DMS object internally and also documents externally.

When sending internally enaio® client creates text files with the extension `os`. The file extension `os` is registered by enaio® client.

Saved queries, SQL queries, and links to external programs can be sent internally as an OS file. A recipient who with enaio® client running can transfer these into the object search and execute them.

The data are sent in the OS file in the same way they are stored in the table `osconf`.

Links to documents, registers, folders and portfolios can also be sent internally as OS files. A recipient with access to enaio® client opens the DMS object by clicking on the OS file. Access rights apply.

Example of an OS file with a link to a document:

```
[OSAS]
DOKUMENTTYP=Fax
SCHRANK=Adressen
[Fax]
#OSID#=1436
[Adressen]
#OSID#=250
```

If the document is sent externally, enaio® client generates the files and attaches them to the e-mail. The naming schema corresponds to the one used for exporting documents from hit lists and folder windows.

Sending documents, registers and folders is logged in the table `osobjhist` and displayed in the editing history.

Printing

To print data sheets, hit lists and e-mail, enaio® client creates XML files, formats them using style sheets into HTML files and prints them.

The files are saved inside the user-specific WORK area `temp\ostemp` and deleted when the client is quit. The style sheets are located in the directory `clients\client32`.

Hit lists are formatted with the style sheet `print1.xslt`, data sheets with `print2.xslt` and e-mails with `print3.xslt`.

When printing data sheets, data of the dialog element 'TABLE' and data of multi-fields are not printed. If objects have multiple locations, only the location is printed which is displayed on the tab. Documents are moved to the CACHE area before printing.

W-documents are printed out of the W-application. XML documents are just as for display converted into HTML files using style sheets and then saved and printed. Video documents cannot be printed. Image documents can be printed from hit lists and folder windows. Annotations on layers which the user is not permitted to alter, are burnt in before moved to the CACHE area.

Different user-specific settings are managed for printing black/white images and color or grayscale images.

The data are saved in the table `osconf` with configuration type `3` data.

The data for black/white print are saved in the section [ASD], for color and grayscale print in the section [ASP]. Annotations which can be edited by the user are not printed.

If an image document is open, another print function is available. This print function allows you to print annotation which user are allowed to alter. The print dialog is opened with the keyboard shortcut Alt-P. The printing process will consume more time, though.

The printing of documents is logged in the table `osobjhist` and displayed in the editing history.

XML Data

XML data are formatted with style sheets located in the directory `clients\client32`. You can edit the style sheets with any editor.

XML Data for Hit Lists

The style sheet `print1.xslt` formats XML hit lists.

Example of an XML hit list:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <!-- optimal systems GmbH -->
 <Listen>
 <Trefferliste>
 <Anfrageliste>
 <Zeile>
 <DFeld>Adressen</DFeld>
 <DFeld>Kurzname = dros</DFeld>
 </Zeile>
 </Anfrageliste>
 <Datenliste>
 <Zeile>
 <HFeld>Kurzname</HFeld>
 <HFeld>Name</HFeld>
 <HFeld>Vorname</HFeld>
 </Zeile>
 <Zeile>
 <DFeld>Drossel</DFeld>
 <DFeld>Drossel-Beyer</DFeld>
 <DFeld>Paul</DFeld>
 </Zeile>
 <Zeile>
 <DFeld>Drosse</DFeld>
 <DFeld>Drosse-Meierstein</DFeld>
 <DFeld>Paula</DFeld>
 </Zeile>
 </Datenliste>
 <Allgemein>
 <Titel>Dokumentliste</Titel>
 <Benutzer>THOMAS</Benutzer>
 <Dokumenttyp>Adressen</Dokumenttyp>
 <Anzahl>2</Anzahl>
 </Allgemein>
 </Trefferliste>
```

XML data for hit lists contain the section <Anfrageliste> with the search criteria, the section <Datenliste> with the hit list data and the section <Allgemein> with information on the 'Anzahl der Treffer' (number of hits), the 'Benutzer' (user/s), 'Titel' (title), and the 'Objekttyp' (object type).

The file list contains the indexing data displayed in the columns in the hit list.

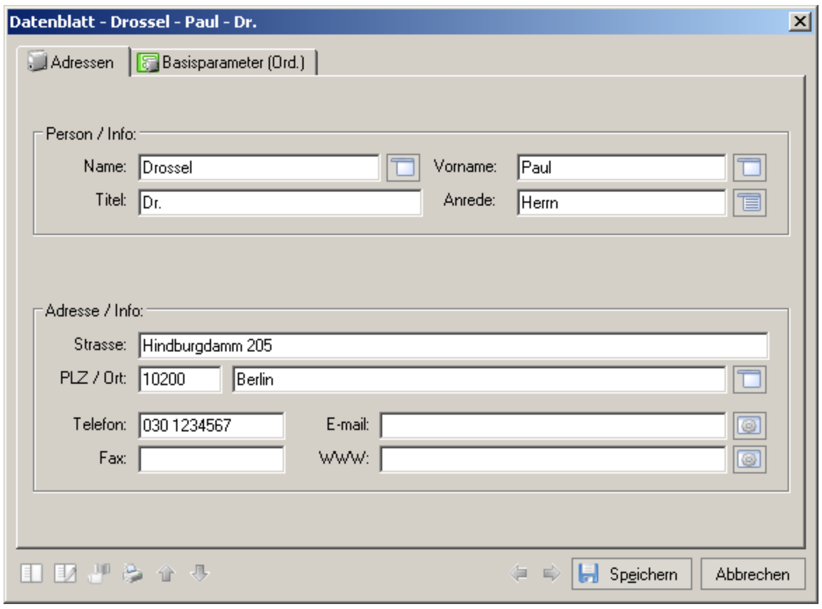
XML Data for Data Sheets

The style sheet `print2.xslt` formats XML data sheets.

Example of an XML data sheet:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <!-- optimal systems GmbH -->
 <Datenblatt>
 <Allgemein>
 <Titel>Datenblatt</Titel>
 <Benutzer>THOMAS</Benutzer>
 </Allgemein>
 <Karten>
 <Karte>
 <Dokumenttyp>Adressen</Dokumenttyp>
 <Zeile>
 <DFeld>Name</DFeld>
 <DFeld>Drossel </DFeld>
 </Zeile>
 <Zeile>
 <DFeld>Vorname</DFeld>
 <DFeld>Paul</DFeld>
 </Zeile>
 <Zeile>
 <DFeld>Strasse</DFeld>
 <DFeld>Hindburgdamm 205</DFeld>
 </Zeile>
 <Zeile>
 <DFeld>Titel</DFeld>
 <DFeld>Dr. <DFeld/>
 </Zeile>
 <Zeile>
 <DFeld>Telefon</DFeld>
 <DFeld>030 1234567</DFeld>
 </Zeile>
 <Zeile>
 <DFeld>Wohnort</DFeld>
 <DFeld>Berlin</DFeld>
 </Zeile>
 <Zeile>
 <DFeld>PLZ/Ort</DFeld>
 <DFeld>10200</DFeld>
 </Zeile>
 <Zeile>
 <DFeld>Fax</DFeld>
 <DFeld />
 </Zeile>
 <Zeile>
 <DFeld>E-mail</DFeld>
 <DFeld />
 </Zeile>
 <Zeile>
 <DFeld>Anrede</DFeld>
 <DFeld>Herrn</DFeld>
 </Zeile>
 <Zeile>
 <DFeld>WWW</DFeld>
 <DFeld />
 </Zeile>
 </Karte>
 </Karten>
 </Datenblatt>
```

The following folder data sheet was printed:



Person / Info:

Name: Drossel
Vorname: Paul
Titel: Dr.
Anrede: Herrn

Adresse / Info:

Strasse: Hindburgdamm 205
PLZ / Ort: 10200 Berlin
Telefon: 030 1234567
E-mail:
Fax:
WWW:

When printing data sheets, the section <Karte> is printer for each displayed tab. The print of a tab data sheet therefore contains the folder section, the print of a document data sheet always contains a folder and a basic parameter section.

XML Data for E-Mails

The style sheet `print3.xslt` formats XML e-mails.

Example of an XML e-mail:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <!-- optimal systems GmbH -->
 <E-Mail>
 <Common>
 <User>THOMAS</User>
 </Common>
 <Data>
 <From>Jürgen Nitz</From>
 <Date>23.10.2002 16:17</Date>
 <To>Entwicklung; Consulting; Support</To>
 <Cc />
 <Subject>Release Warning</Subject>
 <Body><<ole0.bmp>> Betrifft Versionen: Update von 3.60 SPVII auf OS:4x SPIII
Symptome: Ab 3.60 SPVII wird die Datei axindex.cfg aufgesplittet. OS:4x Versionen
können bisher nicht damit umgehen. Lösung: axindex.exe 4.00.569 als Hotfix im SP-
Verzeichnis</Body>
 </Data>
 </E-Mail>
```

Print Labeling for DMS Documents

Print labeling is set up with enaio® administrator. When printing image documents, the configured headers and footers are printed too.

The configuration is saved in the file `as.cfg`.

Example:

```
[PRINTHEADER]
FONT=Arial
FONTSIZE=12
POSITION=2
TEXT=Archiv-Ausdruck
TEXTCOLOR=3
TRANSPARENT=0
```

The configuration is valid for all image documents and all users.

Form Printing

enaio® report-generator can be used to merge index data of different DMS objects, format and print them. The external program `axvbform.exe` is integrated for form printing. The program requires the configuration file `axvbform.cfg` which contains assignments of DMS object types to forms and form sets.

If the program is started with DMS objects, it determines with the configuration file the needed files and requests them via the COM interface from enaio® client. Data to which the user has no access can be accessed via the COM interface. The client does not mind access permission if DMS object types are set to 'Do not enforce the security system' in enaio® editor. The data are printed either as a temporarily generated Microsoft Word document or as a List&Label formatting file. Instead of printing data this way, forms containing the data can be transferred with enaio® printer as a document into the DMS.

enaio® portable und enaio® portableclient

Viewing and editing data on workstations that neither enaio® client nor enaio® webclient is installed on is realized by exporting folders registers or documents from enaio®.

Such data is exported by enaio® portable and viewed and edited with enaio® portableclient. The data is afterwards imported by enaio® portable.

As long as data is exported, other users can only view it in enaio®, but cannot edit it. When importing, modified data is updated in enaio®.

The use of AddOns, events or enaio® data-transfer may result in complex relations between object data in enaio®. Thus, the import of objects may cause data integrity errors.

enaio® portable

The enaio® portable component is integrated with enaio® client as an external program. enaio® portable provides data export configurations, and it allows you to directly export data.

When exporting, enaio® portable creates a database containing all of the necessary data of folders, registers and documents and copies the document files of exported documents.

These data are saved to and managed in a structure that can be accessed with enaio® portableclient.

If these data are accessed in any other way than with enaio® portableclient, data integrity is lost and data cannot be imported again.

enaio® portable imports data back into enaio® that was exported beforehand. During the import, you can specify in detail which changes are meant to be applied and which not.

After import, the enaio® objects can be edited again.

enaio® portableclient

enaio® portableclient is used to access the data which have been exported by enaio® portable. enaio® portableclient resembles enaio® client and allows for similarly systematic data access.

Accessing W-document using enaio® portableclient requires that the necessary are installed at the workstation.

enaio® client provides catalogs and AddOns for indexing objects, whereas in enaio® portableclient these means offer only limited functionality or are not available at all.

As it is part of the enaio® installation data, enaio® portableclient can be installed at the desired workstations.

Editing Documents

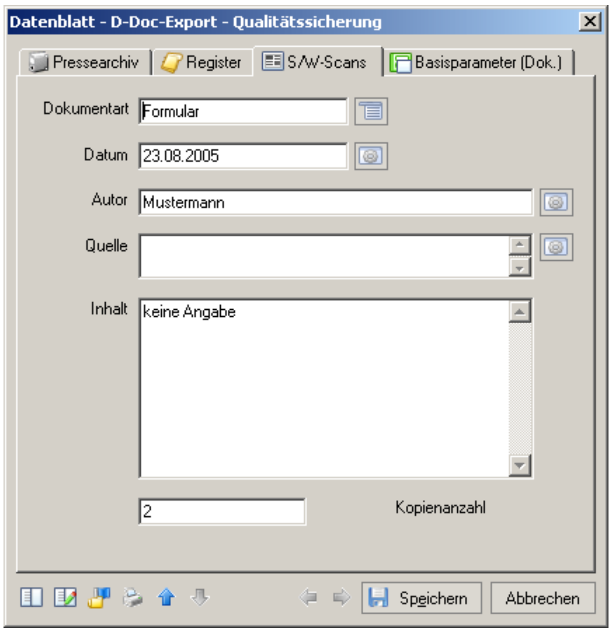
General Requirements

In order to edit documents, enaio® client requires the current object definition, object relations and configuration entries which it loads at startup. It also requires the current rights of the logged-on user. The rights information can be updated in enaio® client with the shortcut Shift + Ctrl + F5.

Changes to the indexing of DMS objects can be permitted or restricted in great detail via the rights system and the properties of dialog elements. Document files cannot be changed once they were archived. Editable copies of archived documents (also variants of Windows documents) can be created, though. Changes to DMS objects can be saved, viewed and undone in the editing history. The deletion of DMS objects cannot be undone.

Changing the Indexing

The indexing of a folder, register, or document is changed via the data sheet. Data sheets of register contain an additional folder indexing tab, data sheets of documents contain an additional tab for the basic parameters of the document and eventually a register indexing tab.


Datenblatt - D-Doc-Export - Qualitätssicherung

Dokumentart: Formular
Datum: 23.08.2005
Autor: Mustermann
Quelle:
Inhalt: keine Angabe
Kopienanzahl: 2

Figure 24

If a user has access to a register but not to the folder in which the register is located, the data sheet of the register will not contain the folder indexing tab.

Basic parameters are automatically executed and cannot be edited. All other fields on tabs of a data sheet can be edited if this is permitted by the rights system or the properties of the dialog elements. Not editable dialog elements are grayed out. If a user opens the data sheet of a document, he can also change the indexing of the folder and the register on the folder tab.

Data sheets can be edited even if other users have opened the same data sheets and edit them. Changes of the indexing are logged by the client in the table `osobjhist`.

The table has the following structure:

Column	Contents
`osguid`	the unique ID of each action
`osid`	the unique ID of the object the action refers to
`ostype`	object type ID
`osuser`	the unique ID of the user who has executed the action
`ostime`	the time stamp
`osaction`	an action ID
`osstation`	the GUID of the workstation on which the action was executed
`osinfo`	the info text which is displayed in the editing history next to the action description

Table 30

The action ID for changes to the indexing is `3`.

Logging of these actions takes place independently from OXRPT logging.

The user who changed the indexing is also entered in the `modifyuser` column of the index table, the time in the column `modifytime`. A system-wide confirmation dialog for saving the indexing can be activated in enaio® administrator. A confirmation dialog is also displayed when creating DMS objects, but not when switching between tabs. This system-wide setting in saved in the file `as.cfg`.

Index data history

If in enaio® editor the property 'Create index data history' was assigned to a DMS object type, the entire existing indexing is transferred into the shadow table.

Shadow table have the same names as the indexing tables, but with an appended 's'.

The shadow tables have the following structure:

Column	Contents
`osguid`	the unique ID of the change
`field#`	the indexing of the object in the individual fields before the change

Table 31

All further data of the change are entered in the table `osobjhist` and are there also labeled with the unique ID (column `osguid`). Changes can be viewed and individual states restored in the editing history.

Folder, Register and Document Indexing

The indexing is changed via the data sheet. If users enter data which do not meet the specifications of the indexing, the dialog elements are flagged and the user is requested to correct the entries. If the indexing matches the specifications, the data are saved in the database.

If the specifications were changed after the creation of the DMS object, already saved indexing may be required to be adapted to the modified specifications to perform further changes.

Users can enter data which affect the rights system. Example: a user enters '300000' in the 'purchase volume' field, but he does not have access to objects which are indexed with values of 300000 and above via expressions. As soon as he saved the indexing, the client updates the view. The object is no longer visible for the user.

AddOns with the property 'Execute automatically' will be automatically executed once the indexing is saved. When using the Quickfinder AddOn or the Filing plan AddOn, possible changes to the entries of connected index fields will not be checked.

If a user changes the indexing on one tab of the data sheet and switches to another tab, the indexing is automatically saved in the database or a confirmation dialog displayed before the switch. This user-specific setting on the 'CONFIRM' tab is saved in the table `osconf` along with configuration type `3` data. Even if a system-wide confirmation dialog is switched on for saving the indexing, a confirmation dialog when switching between tabs is only displayed if the user has explicitly turned this setting on.

The user who changed the indexing is also entered in the `modifyuser` column of the index table, the time in the column `modifytime`. The action is entered in the table `osobjhist`. The previous index data history is entered in the shadow table.

Modification data of folders and registers are displayed via the 'Object information' function. Modification data of documents are also displayed on the 'BASIC PARAMETERS' tab of the data sheet.

If the DMS object is subscribed to, the client enters the object's data in the table `osrevisit`.

Changes in Batch Mode

A batch change is the simultaneous change of the indexing of several DMS objects of the same type. In case of batch changes enaio® client does not verify if the data entered by the user meet the indexing specifications. It is also not verified if the index fields are write-protected. Events are not executed, AddOns are not executed automatically. Only the database type and the permitted length of the entry are verified.

The indexing of multi-fields and the indexing of the dialog element 'TABLE' cannot be changed in batch mode.

Changes performed in batch mode can lead to inconsistent data.

The right to execute changes in batch mode must be explicitly granted in enaio® administrator via the system role 'Client: execute changes in batch mode'. If this system role is not assigned, changes in batch mode are not possible.

Changing the Location

The location of a document is administered independent of the indexing. The location data are stored in the table `sdrel`. It stores the ID of the corresponding folder and the ID of the register of the next upper level.

The location data of registers are saved in the table `osparregrel` and also in the index table `register#`. Both tables store the ID of the corresponding folder and the ID of the register of the next upper level. Folder are located at the top level of the hierarchy and have no location data.

If a user moves a register, the location data are updated in the index table and the location table `osparregrel`. All documents which are in a register will also be moved. The location data of these documents changes if the register is moved into another folder. Then, the location data in the table `sdrel` are also updated.

If a document is moved, the location data in the table `sdrel` are updated. For documents which are moved from the filing tray to a folder, the location data will be removed from the table `mdrel` and the new location is entered in the table `sdrel`.

User must first assign a type to documents without type and index them before they can be moved to a folder. The location and the storage location of the document files are changed. Location changes will not be logged in the editing history.

Use enaio® administrator to prevent the moving of documents or archived documents. This setting is managed in the file `as.cfg`.

Moving can cause data inconsistencies, e.g. in case of paginated documents or for indexing via Quickfinder and the filing plan AddOn.

Multiple Locations through References

Documents can have multiple locations. Documents with multiple locations have multiple entries in the table `sdrel`. Such documents can be accessed from any of the locations equally. The indexing and the document files are managed just as documents with only one location. If a user deletes a document with multiple locations, only the reference pointing to the current location is removed from the table `sdrel`.

The creation of a new location and the deletion of a location are not logged in the table `osobjhist` and are therefore not displayed in the editing history.

A new location can be created by holding down the CTRL key and clicking with the left mouse button and dragging the document into a different folder or register or by clicking the document with the right mouse button and dragging it to a different folder or register and choosing 'COPY TO HERE' from the context menu.

Creating references can be disabled in enaio® administrator. This setting is managed in the file `as.cfg`. Folders and registers only have one location. If multiple references exist for one document, the count is displayed on the folder and register tabs of the document's data sheet. A function of the tabs allows you to switch between the index data. Creating references will not be logged in the editing history.

Modifying Document Files

Pages can be added of deleted for image documents.

Video documents have the corresponding digital sequence assigned.

Windows documents are opened in the respective application for editing. The file can be edited and saved in this application in any way.

Image documents, video documents and Windows documents opened by a user for editing, are checked out. They are indicated in indexing table `objectn` by the user name (`lockuser`) and entered in the table `doclock`. Other users can open only a read-only copy of the documents.

The table `doclock` has the following structure:

Column	Contents
`object_id`	the unique ID of the DMS object
`object_type`	object type ID
`user_id`	the unique ID of the user who edits the document
	the time stamp
`user_time`	status information
`info`	modification date of Windows Documents
`fileinfo`	user workstation
`station`	

Table 32

When the document is checked in, the user name (`lockuser`) is deleted from the table `object`*n* and the entry is removed from the table `doclock`.

Changes made to the files are logged by the client in the table `osobjhist` and displayed in the editing history.

If the document history is enabled, the server saves the different document versions to the WORK area in a directory which is created on the same level as the document files and is named after the document ID. The versions are named as follows:

```
[osguid]_[DocID].[Extension]
```

[osguid] the unique ID of the change from the table `osobjhist`.

[DocID] the unique ID of the document.

Opening documents for editing and modifying document files will be logged in the editing history.

Modifying Image Documents

If a user opens an image document for editing, it is checked out, moved by the server to the CACHE area of the user and opened in the editing module. Pages can be added or deleted in the editing module.

If a user deletes pages, they are instantly removed from the CACHE area, added pages are transferred to the WORK area. Once the user has saved the changes, the files are transferred to the server and deleted from the CACHE and WORK area. The document is checked in. The server saves all files according to the same schema when creating new files.

If pages are rotated and the document is saved, the rotated page is saved. This change will be logged in the editing history. The original version will also be saved for the document history.

If a user rotates a page, annotations on layers are not automatically rotated for this page.

If a user does not save the changes, the document is checked in, the files are deleted from the CACHE area but not transferred to the server.

Modifying Video Documents

If a user opens a video document for editing, it is checked out, moved by the server to the CACHE area of the user and opened in the editing module.

The user can select a file in the editing module. The file will then be copied to the WORK area. If the user saves the changes, the files are transferred to the server and deleted from the WORK area. The old version is deleted from the CACHE area. The server saves all files according to the same schema when creating new files.

The open for editing action will be logged in the editing history. If a user does not save the changes, the document is checked in, the files are deleted from the CACHE area but not transferred to the server.

Modifying Windows Documents

Windows documents are not edited in an editing module, but in the respective application. If a user opens a Windows document for editing, it is checked out and moved by the server to the CACHE area of the user.

The client determines the application corresponding to the extension with the help of the table `ostemplate` and the path to the application with the table `osapplications` if the application is not supposed to be launched as a registered application. The file will be opened in the application and can be edited by the user. If the user saves the file, it must be saved under the specified name and path. The file must be closed, too. As long as it remains open, the client cannot transfer it to the server and check it in.

If the user closes the document without saving, the document will remain checked out. The client will not check if the document was changed in the application. If the user has closed the file, he can check it in or undo the checkout. If the user checks in the document, it is transferred to the server and deleted from the CACHE area. The data in the index table and the table `doclock` are updated. If the user undoes the checkout, the file is deleted from the CACHE area and the data are also updated in the database.

If a user does not check in a document, but opens it again, it is opened from the CACHE area.

The open for editing action will be logged in the editing history.

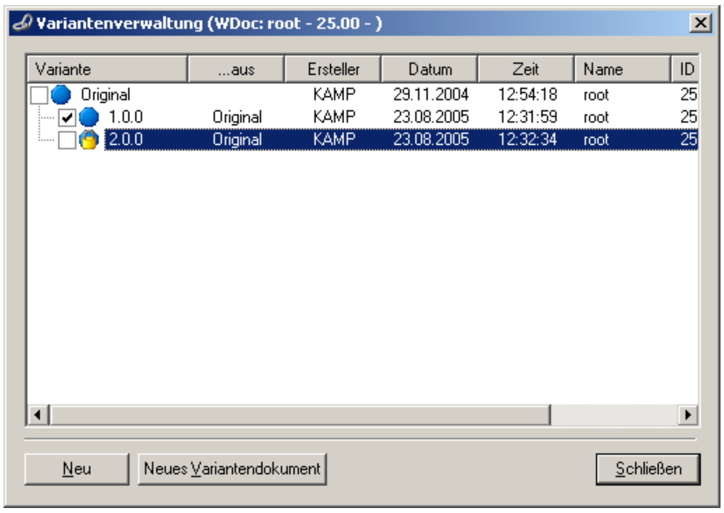
Check can be automated using enaio® office-utilities for Microsoft Office applications.

When quitting the client, all checked-out documents are checked in automatically. If checked-out documents are still open in the application, the user will be requested to close and check in the documents or to undo the checkout.

Variant Administration

Variants can be created for Windows documents.

Variants are copies of copy Windows which have no location and can only be accessed through the variant administration window of the original document.


Variantenverwaltung (WDoc: root - 25.00 -)

Variante	...aus	Ersteller	Datum	Zeit	Name	ID
Original		KAMP	29.11.2004	12:54:18	root	25
✓ 1.0.0	Original	KAMP	23.08.2005	12:31:59	root	25
2.0.0	Original	KAMP	23.08.2005	12:32:34	root	25

Figure 25

If a user creates a variant, a copy of the source document is created and opened. The document must be closed and checked in after saving. Variants of variants can be created. Each variant has its own editing history.

This variant will have the same indexing as its source document. The indexing is saved in the table `object#`. It can be changed, but not queried. The Windows document of which the first variant is created, will have the entry `'1'` in column `versions` of the indexing table `object#`. It is indicated as the original document in the variant administration. Only the original can be queried. The variant administration can only be accessed via the original.

For variants, the unique ID of the original is entered into the column `version` of the indexing table. Variants do not have a location and therefore no entry in the table `sdrel`.

All other data of the variant administration are found in the table `osdocver`.

The table has the following structure:

Column	Contents
`doc_id`	the unique ID of the document
`doc_parent`	the unique ID of the source document
`doc_act`	flag of the current document
`doc_ver`	the name of the variant
`doc_root`	the unique ID of the original

Table 33

For originals a '`0`' is entered in the column `doc_parent`.

In the variant administration a document can be marked as the current document. For this document a '`1`' is entered in the column `doc_act`. A document flagged in such way is opened if the Windows document is opened in a hit list or the folder window. Hit lists and folder windows will always display the indexing of the original.

File management and filing location correspond to those of other documents.

If an original is deleted in a hit list or the folder window, all variants are deleted, too. Variants, but not the original can be deleted in the variant administration window.

Annotations on Layers

You can add annotations to layers of black/white, grayscale and color documents. Creating and editing layers will not be logged in the editing history.

If a user opens a file with a layer which he is not permitted to modify, the layer is burnt into the file before it is moved into the user's CACHE area. The layer cannot be removed. The file is sent, printed and exported this way. If a user is permitted to modify a document file, but not its layer, he will edit the files with burnt in layers in the CACHE area. The files are moved to and saved on the server this ways. This possibility should be restricted with respective settings in the rights system. If a user edits a layer, it is not checked out. It will not be indicated to another user that the layer is being edited and he can edit the layer too. The layer is saved as the last saved version.

Layers are managed in the database in the table `annotations` and `ann-groups`. The table `annotations` contains data, the table `ann_groups` contains group assignments of group layers.

The table `annotations` has the following structure:

Column	Contents
`annotation_id`	the unique ID of the layer
`object_id`	the unique ID of the document
`page`	the page to which the layer is assigned
`user_id`	the user who created the layer
`flag`	the layer type
`zeitstempel`	the exact time when the layer was created
`timeout`	not used
`annotation`	the layers and annotation data

Table 34

The table `ann_groups` has the following structure:

Column	Contents
`gruppen_id`	the unique ID of a group to which the group layer was assigned.
`annotation_id`	the unique ID of the group layer

Table 35

Annotations on layers are not included in the editing history. There, documents are always displayed with the layers of the current version.

Electronic Signature

A card-reading device, a signature card, the PIN and an installation of Adobe Reader and the respective license are required for digital signatures.

enaio® client signs documents in PDF format in conjunction with Adobe Reader. For this signature, a signature type can be set up in enaio® administrator.

Data of the signature types are managed in the database in the table `ossigtext`. The table has the following structure:

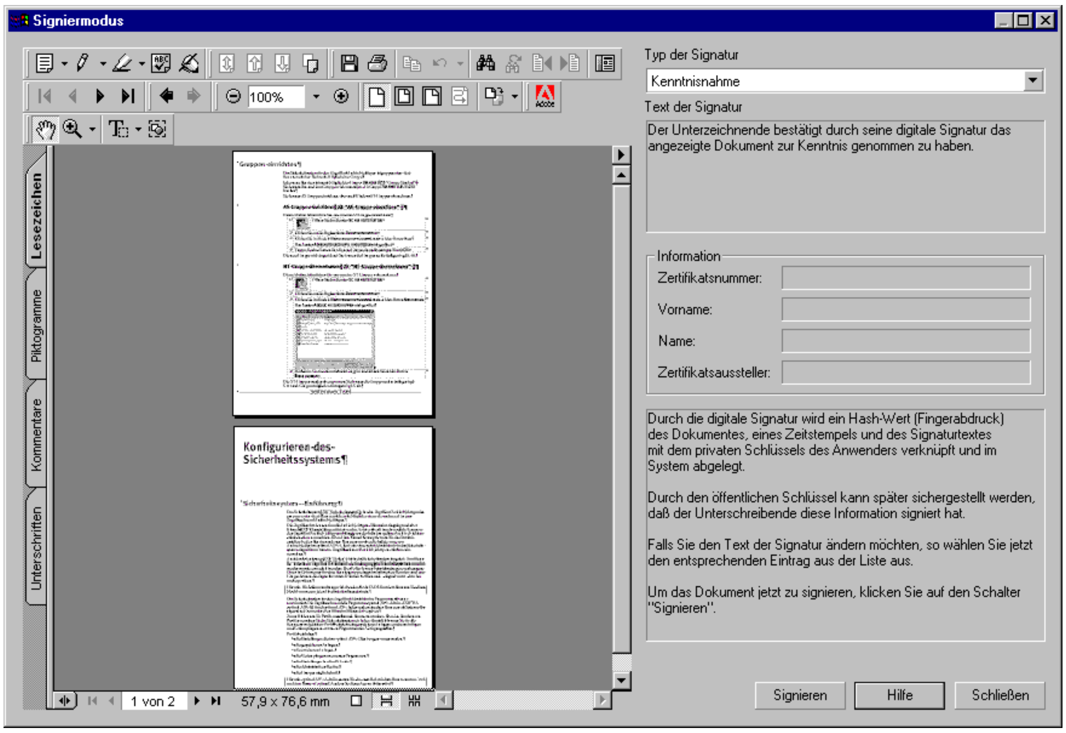
Column	Contents
`ostextid`	the unique ID of the signature text
`oslabel`	a short description of the signature text
`ostext`	the signature text

`osdeleted`	Information, if the signature text is still being used.

Table 36

If a user wants to sign a document digitally, he selects the document and starts the digital signature via the context menu or with the shortcut Ctrl+S. The files are then moved to the CACHE area. Annotations on layers which the user is not permitted to alter, are burnt in.

PDF documents are opened with the signature module.



Typ der Signatur: Kenntnisnahme

Text der Signatur: Der Unterzeichnende bestätigt durch seine digitale Signatur das angezeigte Dokument zur Kenntnis genommen zu haben.

Information
Zertifikatsnummer:
Vorname:
Name:
Zertifikatsaussteller:

Durch die digitale Signatur wird ein Hash-Wert (Fingerabdruck) des Dokumentes, eines Zeitstempels und des Signaturtextes mit dem privaten Schlüssel des Anwenders verknüpft und im System abgelegt.

Durch den öffentlichen Schlüssel kann später sichergestellt werden, daß der Unterschreibende diese Information signiert hat.

Falls Sie den Text der Signatur ändern möchten, so wählen Sie jetzt den entsprechenden Eintrag aus der Liste aus.

Um das Dokument jetzt zu signieren, klicken Sie auf den Schalter "Signieren".

Figure 26

The user chooses the signature type and confirms the electronic signature with his signature card and PIN.

The signature is then saved in the created PDF file.

The PDF file is saved just any other version on the server in a directory which is created on the same level as the document file and is named according to the document ID. The file itself receives as any other version a unique ID of the change in table `osobjhist` followed by the unique ID of the document as its name.

Electronic signatures on PDF files can be verified in the editing history. During the verification the PDF file is opened in the signature module and the signature text and header stored in the database are displayed. The signature is decrypted with the public key. IT is verified if the hash value of file, signature text and signature header matches the hash value in the database.

If a document is deleted which has a corresponding digitally signed version, the digitally signed version is deleted, too.

No signature types are available for the electronic signature of W-Documents, e-mail document and XML documents. An external signature – a PKCS7 file – is created and saved together with the signed document in the WORK area for documents of this document type.

A document which is signed cannot be displayed at the same time. The user will then receive a notification.

External signatures can also be verified in the editing history.

An entry in the file `as.cfg` is used to set the algorithm - SHA-256, SHA-386 or SHA-512.

Deleting Objects

The access permissions for folders, registers and documents in enaio® administrator dictate the deletion permissions. Just like other access rights, deletion permissions can depend on the indexing via logical expressions. User-specific confirmation dialogs can be enabled for the deletion of DMS objects. Objects deleted by the user are moved to the trash can. Every user has its own trash can.

Confirmation dialog settings on the '**CONFIRM**' tab are saved in the table `osconf` with configuration type `3` data and can be distributed using the profile administration.

When deleting folders and registers, the entire contents are deleted recursively. Therefore, the user requires deletion rights for all contained objects. Archived documents cannot be deleted. If folders or registers contain archived documents, these must be moved first.

Documents which are checked out can be deleted by any user. Independently of the confirmation dialog settings the user will be informed which user has opened the document for editing when and at which workstation and must confirm to delete.

In a further step, objects can be deleted from the trash can. For this purpose a system trash can exists which contains the trash can of all users. Users with the respective permissions can view the trash can and permanently delete objects in the system. When deleting DMS objects, enaio® client deletes all database entries which refer to these objects.

The following tables will be updated:

- the indexing tables `stamm#`, `register#`, `object#`,
- the shadow tables `stamm#s`, `register#s`, `object#s`,
- the location tables `sdrel` and `osparregrel`,
- the table `mdrel`, if objects are managed in portfolios,
- the table `osdigsigs` for signed documents,
- the table `osdocver` for variants,
- the table `remarks` for text notes,
- the table `hyplnk` for references,
- the tables `annotations` and `ann_groups` for annotations on transparencies,
- the table `doclock`, if documents are checked out,
- the table `osrevisit` for follow-ups and current subscriptions.

For reference documents only the location entry in the table `sdrel` is deleted.

The table `osabonnement` for subscribed objects is updated by enaio® server.

The table `osdigsigs` for signed documents is not updated.

Text notes and document files, versions, signed versions and variants are deleted from the server. And all files which were created for full text indexing.

Files are deleted from the CACHE area when the client is quit.

It is not checked if references to deleted objects exist in the object search or on the search bar. It is also not checked if the indexing of deleted objects was linked to the indexing of other objects.

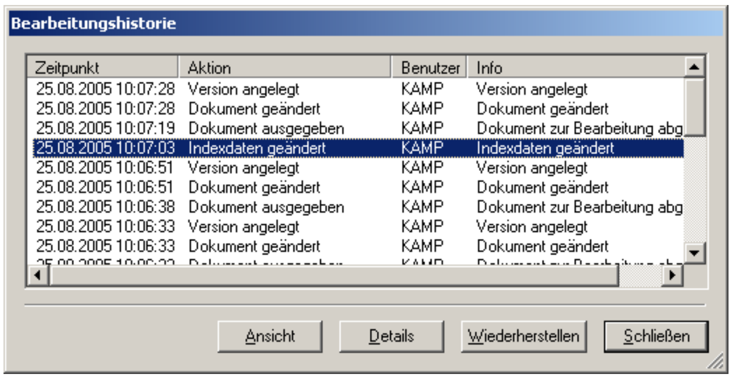
Editing history

The editing history of an object displays entries for the object stored in the table `osobjhist` and allows index versions and file versions for documents to be restored if object types have their properties set accordingly in enaio® editor. Histories can be enabled/disabled globally in enaio® editor, although another setting allows a user (with the respective system role a button is displayed in the properties window of a document) to enable/disable both histories at any time.

The signature can be verified in the editing history for digitally signed documents.

Restoring the Indexing

Changes of indexing are logged in the table `osobjhist` with the action ID `3` and are indicated in the editing history with the entry 'Index data created', time and user are listed.



Zeitpunkt	Aktion	Benutzer	Info
25.08.2005 10:07:28	Version angelegt	KAMP	Version angelegt
25.08.2005 10:07:28	Dokument geändert	KAMP	Dokument geändert
25.08.2005 10:07:19	Dokument ausgegeben	KAMP	Dokument zur Bearbeitung abg
25.08.2005 10:07:03	Indexdaten geändert	KAMP	Indexdaten geändert
25.08.2005 10:06:51	Version angelegt	KAMP	Version angelegt
25.08.2005 10:06:51	Dokument geändert	KAMP	Dokument geändert
25.08.2005 10:06:38	Dokument ausgegeben	KAMP	Dokument zur Bearbeitung abg
25.08.2005 10:06:33	Version angelegt	KAMP	Version angelegt
25.08.2005 10:06:33	Dokument geändert	KAMP	Dokument geändert

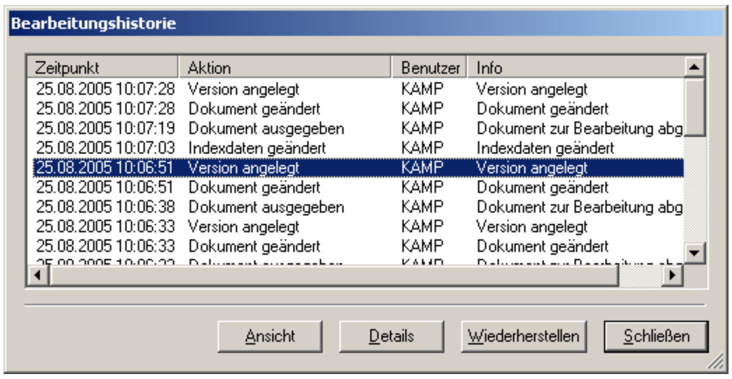
Figure 27

If this entry is selected, the indexing before the modification can be viewed with the **View** button. It is loaded from the shadow table. Use the **Restore** button to edit the indexing from before the change on the data sheet. It can be edited and saved.

When saving, the indexing is entered in the indexing table, the indexing from before the change is entered in the shadow table. The change in entered in the table `osobjhist` with the action ID `3` and the information (`osinfo`) 'Recovery of index data dated 'date' 'time''.

Restoring Versions

Changes of document files are logged in the table `osobjhist` with the action ID `4` and are indicated in the editing history with the entry 'Version created', time and user are listed.



Zeitpunkt	Aktion	Benutzer	Info
25.08.2005 10:07:28	Version angelegt	KAMP	Version angelegt
25.08.2005 10:07:28	Dokument geändert	KAMP	Dokument geändert
25.08.2005 10:07:19	Dokument ausgegeben	KAMP	Dokument zur Bearbeitung abg
25.08.2005 10:07:03	Indexdaten geändert	KAMP	Indexdaten geändert
25.08.2005 10:06:51	Version angelegt	KAMP	Version angelegt
25.08.2005 10:06:51	Dokument geändert	KAMP	Dokument geändert
25.08.2005 10:06:38	Dokument ausgegeben	KAMP	Dokument zur Bearbeitung abg
25.08.2005 10:06:33	Version angelegt	KAMP	Version angelegt
25.08.2005 10:06:33	Dokument geändert	KAMP	Dokument geändert

Figure 28

If this entry is selected, the version before the modification can be viewed with the **View** button. The version is transferred to the WORK area `temp\ostemp` and opened as read-only.

To save the current document as a version and save the non-edited version as the current document, press the **Restore** button.

The change in entered in the table `osobjhist` with the action ID `4` and the information (`osinfo`) 'Recovery of document dated 'date' 'time''.

Full Text Search

Introduction

enaio® offers the possibility to index documents using indexing services and to search them in enaio® client through the integrated full text index.

New documents can be indexed upon creation and existing documents can be indexed by an automatic action at a later time.

The contents and the indexing of documents can be indexed. You can also index the indexing of folders and registers.

In combination with the text recognition software FineReader image documents, in particular D-Document types, can be transformed into ASCII files with the help of the text recognition to be then indexed.

enaio® supports the following indexing services:

- Microsoft SQL Server

  Microsoft SQL Server 2008 R2 or 2012 can be used as a full text database.
- Lucene

  Lucene is a full text database completely based on Java that can also be used under Linux. It supports the combination with IntraFind components.
- RetrievalWare

  For more information on RetrievalWare please contact our consulting and support department.

Detailed information can be found in the 'enaio® fulltext' handbook.

Microsoft SQL Server

If you use the database system Microsoft SQL Server with enaio®, you can also use the Microsoft SQL Server as a full text database.

When setting up the Microsoft SQL Server, appropriate configurations are necessary.

The required tables for full text data are set up when installing enaio®. Data for the creation of a full text index are generated by enaio® rendition-plus. For the indexing of image documents with enaio® rendition-plus, text recognition is used.

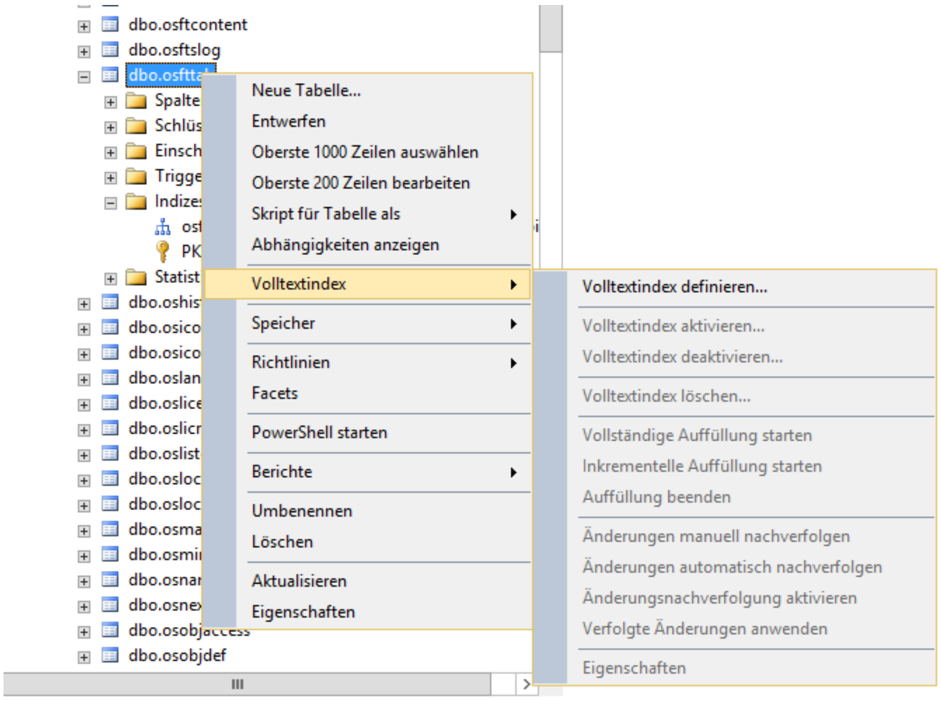
To integrate the Microsoft SQL Server, specify it as the full text database in enaio® enterprise-manager.

Customizing the Full Text Database

When installing enaio®, database tables are created automatically. The `osfttab` table must be configured as a full text table in order to integrate Microsoft SQL Server as a full text database.

To do so, select the `osfttab` table in Microsoft SQL Server and click **FULL-TEXT INDEX › DEFINE FULL-TEXT INDEX** from the context menu.

The Full-Text Indexing Wizard helps you with the configuration.



In the SELECT AN INDEX dialog, you can apply the ready configured index for the table.

Lucene

For full text search enaio® provides an interface to the free and high-performance full text database 'Lucene'. It is completely based on Java and can also be used under Linux.

Its operation is similar to the one of the Microsoft Indexing Service. A directory is managed into which enaio® server writes the data to be indexed. The directory is monitored, incoming data are processed, and the results are written to the full text database.

When issuing a full text query, this full text database is queried.

Windows platforms allow the integration of iFilters for file content extraction.

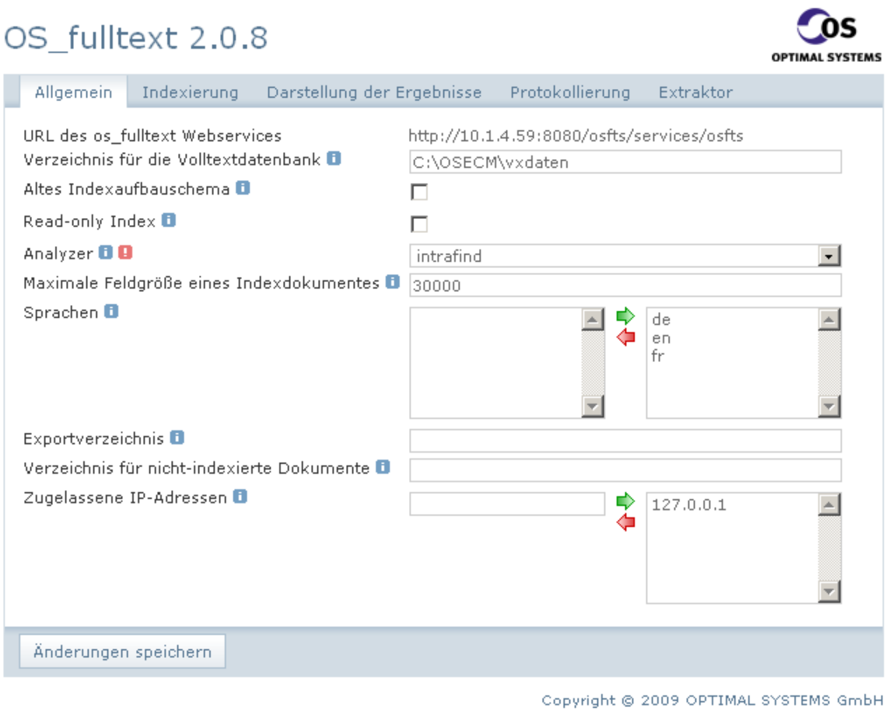
You can also implement functions based on IntraFind technologies:

- 'Did you mean'; a function offering other terms in case of typing errors.
- A list of terms being frequently entered together with search terms that can be added to new queries.
- The summary function which creates and displays summaries out of text documents.
- The function 'Find similar objects' uses the full text database's data to determine similar objects.
- Language analysis. The language of texts is determined before indexing and then analyzed and indexed.

IntraFind requires you to license the 'LIS' module.

enaio® portableclient also supports enaio® fulltext with Lucene. Full text data can be included in data export for enaio® portableclient. These data are made available for searches in enaio® portableclient.

enaio® fulltext with Lucene is a Web archive. The Web application is configured on a configuration page.



OS_fulltext 2.0.8

Allgemein | Indexierung | Darstellung der Ergebnisse | Protokollierung | Extraktor

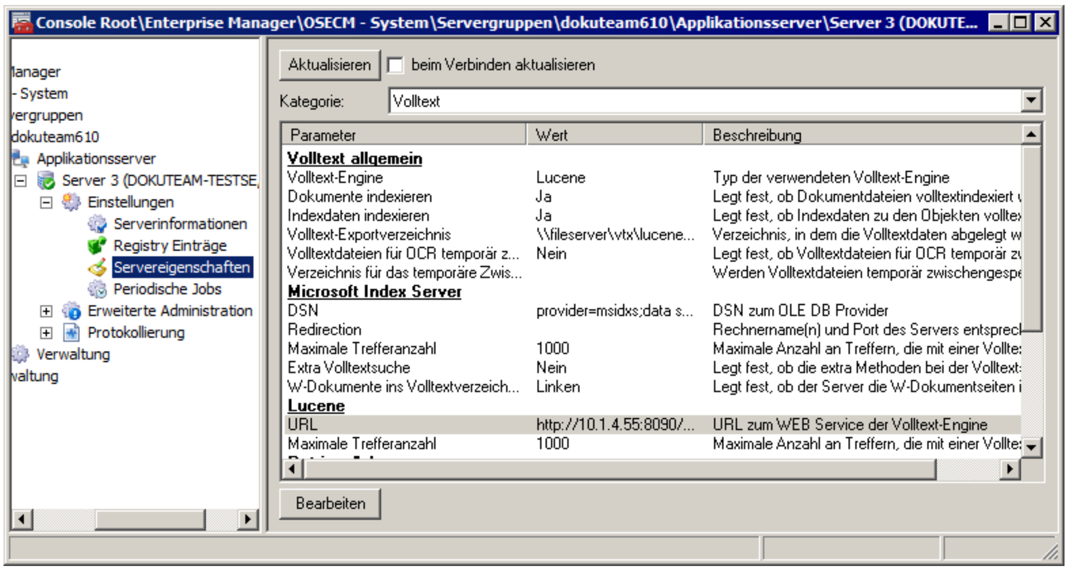
URL des os_fulltext Webservices	http://10.1.4.59:8080/osfts/services/osfts
Verzeichnis für die Volltextdatenbank	C:\OSECM\vxdaten
Altes Indexaufbauschema	☐
Read-only Index	☐
Analyzer	intrafind
Maximale Feldgröße eines Indexdokumentes	30000
Sprachen	de en fr
Exportverzeichnis	
Verzeichnis für nicht-indexierte Dokumente	
Zugelassene IP-Adressen	127.0.0.1

Änderungen speichern

Copyright © 2009 OPTIMAL SYSTEMS GmbH

Figure 29: enaio® fulltext with Lucene: configuration page

Further configuration is done in enaio® enterprise-manager.



Console Root\Enterprise Manager\OSECM - System\Servergruppen\dokuteam610\Applikationsserver\Server 3 (DOKUTE...

lanager
- System
ergruppen
dokuteam610
Applikationsserver
Server 3 (DOKUTEAM-TESTSE,
Einstellungen
Serverinformationen
Registry Einträge
Servereigenschaften
Periodische Jobs
Erweiterte Administration
Protokollierung
Verwaltung
valtung

Aktualisieren ☐ beim Verbinden aktualisieren

Kategorie: Volltext

Parameter	Wert	Beschreibung
Volltext allgemein		
Volltext-Engine	Lucene	Typ der verwendeten Volltext-Engine
Dokumente indexieren	Ja	Legt fest, ob Dokumentdateien volltextindexiert
Indexdaten indexieren	Ja	Legt fest, ob Indexdaten zu den Objekten volltex
Volltext-Exportverzeichnis	\\fileserver\vtx\lucene...	Verzeichnis, in dem die Volltextdaten abgelegt w
Volltextdateien für OCR temporär z...	Nein	Legt fest, ob Volltextdateien für OCR temporär zu
Verzeichnis für das temporäre Zwis...		Werden Volltextdateien temporär zwischengespe
Microsoft Index Server		
DSN	provider=msidxs;data s...	DSN zum OLE DB Provider
Redirection		Rechnername(n) und Port des Servers entsprech
Maximale Trefferanzahl	1000	Maximale Anzahl an Treffern, die mit einer Vollte
Extra Volltextsuche	Nein	Legt fest, ob die extra Methoden bei der Volltext
W-Dokumente ins Volltextverzeich...	Linken	Legt fest, ob der Server die W-Dokumentseiten i
Lucene		
URL	http://10.1.4.55:8090/...	URL zum WEB Service der Volltext-Engine
Maximale Trefferanzahl	1000	Maximale Anzahl an Treffern, die mit einer Vollte

Bearbeiten

Figure 30: enaio® fulltext with Lucene: configuration in enaio® enterprise-manager

Security

Overview

In an ideal situation, DMS to some extent administer data and documents that do not exist in paper form anymore. That is why storage security and availability of digital data is of major importance.

Digital documents are exposed to the following risks:

- Hardware defects, software malfunctions, or external attacks may damage or destroy the DMS and digitally managed data making data restoration impossible. However, there are mechanisms available
  - for restoration of data and document files (database and WORK directory backup),
  - and for DMS recovery (program directory backup).
- Furthermore, document files and index data might be falsified as well as the DMS offers functions to delete and remove data. Nevertheless, the following options reduce these risks:
  - Unique identification of document files (electronic signature)
  - Recovery of earlier object states (object history)
- Digitally administered objects can be accessed by authorized users only and unauthorized persons cannot even see the objects. Unauthorized persons may be interested in such objects. That is why they must be protected from unauthorized external reproduction. For this purpose, the following protective mechanisms are available:
  - Encapsulation of areas containing document files (server architecture)
  - Encryption of document files
  - Security on the network, domain and operating system level
  - Authentication at access of the DMS
  - Authentication at access of the database
- Another situation that should not happen is internal and correctly authorized users seeing objects they should not see. The following mechanisms are available:
  - User authentication definable for each object
  - Modification of access rights with logical expressions

Data security can be improved through mechanisms outside of the enaio® system. Nevertheless, many internal system functions are available in order to ensure the highest data security within the enaio® system as possible.

Data and Document File Protection

Security on the Network, Domain and Operating System Level

Further external methods that are not affected by the DMS include the backup of the system environment in which the enaio® system is embedded. In particular, these are the network, the domain, and the individual workstations on which enaio® components are working. External methods will be described briefly as detailed information can usually be found in relevant books and references.

You must provide security for the existing network using firewalls that monitor the data traffic between the network and the Internet as well as a firewall and anti-virus software installed centrally or on each workstation. The organization of workstations in domains having a central user administration is another important security method. It will make it harder for unauthorized users to access forbidden areas, such as the WORK area of an application server.

The most important condition is a correctly set up user administration in which users are compelled to enter secure passwords. The same applies to user accounts on single servers and workstations that must be protected sufficiently.

Security provided by the enaio® System

Encapsulation of server data areas

Folder sharing is the most frequently applied method that enables collective access to files within networks. This high-maintenance method is risky and very time-consuming as the directory access must be enabled individually for each folder. However, the encapsulation of data areas on application servers provides more security.

Important data areas are the WORK area, the CACHE area, and the archive media. All of these areas might be locally present on a single application server. The WORK area and the CACHE area are subdirectories of the server directory, and the jukebox with the locally connected archive media can be addressed over a drive shortcut. This combination does not require folder sharing. The application server enables the access to document files over the shared IP port. All document files are made available by the application server.

As soon as a second application server has been integrated into the same server group but has been installed on another computer, folders must be made accessible for the WORK and the server area, and each DMS jukebox must be shared in enaio®. But these shares are limited to the user account the application server services are started with. The share limited to the user account reduces security risks and simplifies administration.

Detailed information on encapsulation of the server data areas can be found in the chapter 'System architecture' and in the enaio® server handbook.

Encryption of documents

For each enaio® document type you can set up the encryption of document files. To do so, you just need to configure the encryption property of the respective object type in enaio® editor.

Document files are encoded with symmetric encryption. At first, documents are encrypted and then transferred to the application server that files them into file areas. The application server is also able to encrypt transferred document files with a symmetric encryption method. Both encryption types use separate keys and can be applied either together or independently of each other.

If an authorized user wants to see or edit an encrypted document, it will be transferred in code from the application server to the client application and then decoded and displayed by the client.

This ensures a constant encrypted document transfer across the network. If attackers get unauthorized access to network communication, they will not be able to access the transferred documents.

License system

Although not meant for, the enaio® license system positively affects data security. Licensing allows application servers to be run on server computers definitely identified by GUID only. The actual container file providing licenses is encrypted. Both facts make it very hard if not impossible to reproduce a specific enaio® system.

In addition, the license system can be configured to identify the workstations through GUID that client applications are run on. Every time a client application is launched, the application server checks whether the workstation the application runs on is authorized properly. If it is not, the license is denied. As a result, communication with the application server is not allowed. Hence, the license system works as a security mechanism for it is an authentication system compelling the application login to the application server.

enaio® authentication

enaio® offers a security system sophisticated access permissions to object types and applications can be distributed with. System roles represent application rights and are assigned to individual users. Rights for object types are assigned to using user groups.

Users

Every user must be set up individually unless users are imported from an LDAP directory service. Each user receives a name, a preset password and individual properties. Properties include the assignment to user groups, the configuration of system roles, and whether or not the user should serve as profile user.

In case users are imported from an LDAP directory service, passwords are not administered in the enaio® system but in the LDAP service. Further properties, such as the assignment to user groups, are still set up in enaio®.

User profiles

You can define a created user as user profile and assign it to new users. When assigning a user to a user profile, profile user settings will be transferred to the new user. This includes, e.g. system roles, group membership, and settings in enaio® client and enaio® capture.

System roles

System roles settings include the configuration of application rights in the enaio® applications, enaio® administrator, enaio® start, enaio® editor, enaio® capture, and enaio® client. In enaio® client, system roles are used to control whether or not a user is allowed to edit notes or to view editing histories, for example.

User groups

Users can be collected in user groups. With user groups you can control and distribute object rights. A user cannot access DMS objects if he is not assigned to any user group. That might be useful for users that should never edit DMS objects but execute administrative tasks only. Users can be members in more than one group. Such users receive the access rights to objects from all assigned user groups always taking mostly extensive rights.

Object rights

There are numerous rights for each created object type and its objects. All objects have the same rights: Show index data (R), Write index data (W), Delete object (D), Output object (X), and Write object (U). Documents additionally have the rights Group annotations (G) and Public annotations (P).

These object rights allow you to individually control the user access to DMS objects.

Object rights can be set up for objects types only. For single DMS objects it is not possible to set up rights. So these methods do not allow you to create private objects. However, there are logical expressions available for the creation of private objects, which are described below.

Logical expressions

You can formulate logical expressions for every single access right of an object type with which rights can be restricted. For example, if you formulate a proper logical expression for each object right, the object creator can see and edit the object only. The respective logical expression would be as follows:

```
#ANLEGER# = #BENUTZER#
```

Locking accounts

Failed logins might indicate an attempted system attack. That is why you can configure how to handle failed logins in enaio®. The following three security levels for the start behavior can be set up in the complete system of enaio® administrator:

- No restriction

  There will not be any reaction on failed logins: the login dialog will be displayed together with the respective indication of incorrect user name or password entry.
- End after 3 failed attempts.

  When launching client applications, the login dialog will not be displayed anymore after three failed login attempts. You need to restart the application. When changing the user in enaio® client, the login is cancelled and the currently logged in user remains logged in. When signing out of enaio® client, the client will exit after three unsuccessful attempts.
- Blocking of user account after 3 failed attempts.

  Blocked user accounts can only be unlocked in enaio® administrator.

Detailed information on the enaio® security system can be found in the enaio® administrator handbook and in the chapter 'Administration' of this System Handbook.

Object Identity

The most important function of a DMS system is to ensure that administered objects (data and document files) are reproducible without having been modified or corrupted. This requires a lot as DMS objects are stored and processed at different places. Objects are created in the working memory of a computer, cached on hard disk, transferred to an application server using LAN, and there, in the WORK area they are written on the hard disk of the server. When opening the object, it needs to go all the way back. At some point, the document files will be archived; i.e. it is saved to optical storage media.

Likewise, the indexing is created in a client, transferred to the server using the network, transferred to the database, and so on. In case of damage, such as a defect server hard disk, the object suddenly exists digitally in a backup file only. This is called 'distributed presence of a DMS object'.

Many requirements needed for the multiple presences of unmodified objects cannot be checked in detail. The working memory must work correctly, the operating system and the client application must administer data perfectly, the network protocol and network hardware must transfer data accurately, hard disks, jukeboxes, and optical media must work correctly, and the database must administer all data without any error.

It is not possible to check these requirements and conditions in detail. As experience teaches, all component types usually perform their tasks properly.

That is why the following describes only functions offered by enaio® for ensuring the object identity. This includes data management within the database, electronic signature, document history, and archiving.

Keeping data in a database

System information and object data are saved in a powerful SQL database system. enaio® supports MS SQL Server, Oracle, and Informix. All these established database systems offer the highest data integrity. However, attention should also still be paid to sufficient authentication and data backup with these database systems. These are measures that are beyond the control of the enaio® system.

It is strongly recommended to not enable database access through a database user no password is necessary for. Such a user account offers unprotected access to all database data. In addition, you must permit secure passwords affording adequate protection due to sufficient length and complexity only.

Document Files: Electronic signature

An electronic signature represents the signature in a paper document confirming the validity of an electronic document.

Technical requirements

If technical requirements are fulfilled, enaio® allows you to sign all documents electronically except XML documents, video documents, and e-mails, and to verify electronically signed documents. A card-reading device, a signature card, the PIN and an installation of Adobe Reader, and the respective license are required for digital signatures. The creation of electronic signature requires as follows:

- Installed PKCS#11 library `pkcs11api.dll` version 2.10 based on the Cryptographic Token Interface Standard PKCS#11
- Smart card reader installed on the system over PCSC

  SignTrust has tested the following card reader for your PKCS#11 library: Utimaco CardMan, Siemens-Nixdorf B1 V4.0 and Siemens B1 Snuggle Desktop. (During the realization period, a CARDMAN 2010 will be applied.)
- Signature card of a TrustCenter (During the realization period, there will be cards of Medizon available.)
- HTMLDOC for the conversion of HTML documents to PDF
- PDF creator for Office to convert Office documents to PDF, alternatively the enaio®°printer

- Installed Adobe Reader, at least version 4.0

Creating electronic signatures

An electronic signature is created as displayed in the following figure:

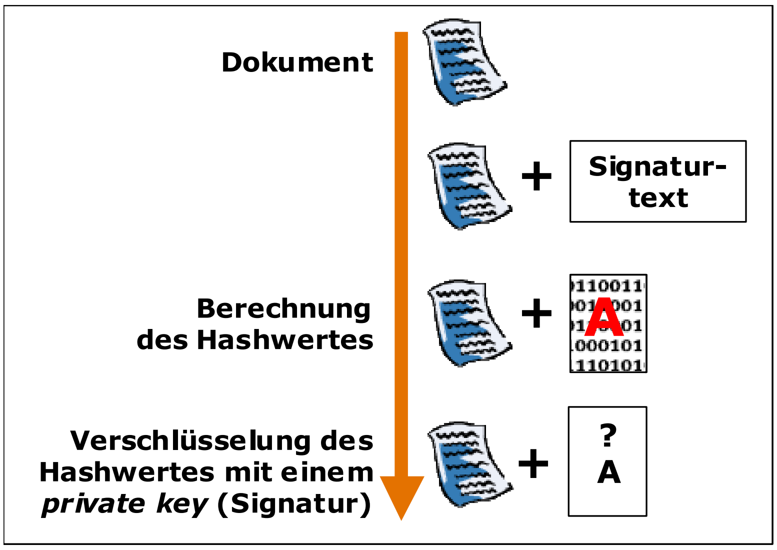

Dokument
+
Signatur-text
Berechnung des Hashwertes
110011 01 001 1 01 000101 110101
A
Verschlüsselung des Hashwertes mit einem *private key* (Signatur)
?
A

Figure 31: Creating electronic signatures

The hash value calculation includes the document and the signature text. The calculated hash value, i.e. a connection of document and signature text will be signed.

The document and the signature type define the signature. Not only is the document signed but also the signature text. The packet to be signed consists of a signature header, a signature text, and the actual document.

- The signature header contains the GUID, a short description and length of the signature text, the user/signer ID, the public key of the user, the certified system location (who has provided the chip card), date, time, workstation number, full name of the user/signer, and optional fields like location, created system etc.
- The signature text is the text deposited in the database.
- Afterwards, the document is attached.

The hash value calculation includes this package and is then signed through the private key. All the told parts together are a document's signature.

Electronic signatures can be directly applied in enaio® client as well as for authorization of workflow processes. In the latter, a document is created automatically at the respective workflow step, it is signed and filed into the workflow file. enaio® client creates a PDF file from the selected document. This PDF file is signed in the signature module and verified.

Database tables

Signature texts and their names are configured in enaio® administrator and filed in the database. Both text and name of a signature are saved in the database table `ossigtext`. So you can open electronic signatures together with a predefined text. This may be useful within workflow scenarios as you can define at which steps the user can deposit which signature type. The signature of a document is saved together with all information in the database table `osdigsig`. The hash value of a document or a document with signature is saved in the database table `osdochash`.

Externally saving and sending electronic signatures

Signatures are saved together with the header and the signature text in a file to enable externally saving and sending.

The signature file and the document allow the user to check whether or not the signature corresponds with the signer. To do so, the user must create a signature package and use the public key of the signer to check whether or not the signature text corresponds with the signature.

Detailed information on how to use electronic signatures can be found in the handbooks of enaio® administrator and enaio® client.

Backup of the OS|mobileDMS App under iOS

Mobile applications are designed to be used on the go and are running on small, portable smartphones or tablets. But because of their size and portability, these devices can easily be stolen by others in a moment of carelessness. When stolen, not only the mobile device is lost, but also the data stored on it. The value of sensitive personal or corporate data is often considerably higher than the monetary value of the mobile device. That is why mobile applications that process sensitive data must ensure the security of these data even in case of theft of the mobile device.

Aside from the actual theft of the mobile device, there is also the possibility of virtual theft of data with the use of malicious apps which were installed on the mobile device out of ignorance or by a third party. These apps often read out sensitive data which are processed on the mobile device and send these data to the developers of the malware.

Security Concept of Apple iOS

Under iOS, the operating system by Apple, all apps are installed and executed separately in different sandboxes. By using sandboxes, the apps are protected from one another under the iOS operating system. Thus, it is impossible for one app to access data of another app. This security concept provided by the operating system already prevents virtual theft of data without taking specific precautions against the installed apps.

In addition, iOS secures data access by prompting the user to enter a password. The user can define a four-digit, numeric password in the operating system that is prompted upon reactivation of the mobile device. Without this password, the device cannot be used. Furthermore, iOS provides an option to delete all data on the mobile device in case the entered password was incorrect ten times in a row. Both security features offer basic protection against theft of sensitive data or in case the mobile device was stolen. This concept depends, however, on the user activating the security features and on the password having a high complexity.

Considerable amounts of data are transferred to mobile devices via wireless networks. The operating system supports the widely-used encryption methods for wireless technologies: WEP, WPA (Enterprise), WPA2 (Enterprise), and VPN. Through these, data connections between the mobile device and the access points are secured. When sending data via a mobile phone network, the A5/3 cipher is used.

Security Concept of OS|mobileDMS

In addition to the security concept provided by the operating system, the OS|mobileDMS app by OPTIMAL SYSTEMS offers further security features.

In order to retrieve data from the enaio® server, the app must connect to the enaio® appconnector REST interface. The Basic Authentication (RFC 2617) method is used which sends the user name and the password as strings in Base64 encoding to enaio® appconnector.

Since the user name and the password are unencrypted when transferred via the HTTP protocol using this authentication method, OS|mobileDMS additionally supports the HTTPS protocol. If the HTTPS protocol is used for the transfer, the entire data communication between OS|mobileDMS and the enaio® appconnector REST interface is encrypted with SSL. The use of SSL is strongly recommended, because the encryption of the data connection via WIFI or UMTS only lasts until the respective access point is reached, and subsequently the data will be forwarded without encryption. SSL ensures full encryption of the connection. For infrastructure elements within this connection, it is difficult to decrypt the data.

Access data used for the authentication of the user at the enaio® server are saved in an unencrypted way within OS|mobileDMS; however, the sandbox of the operating system prevents other apps from accessing this data. Due to the fact that the password is masked in OS|mobileDMS, it is impossible for a third party to see the password in clear text on the mobile device later.

In case of a successful authentication of OS|mobileDMS with enaio® appconnector, all data received by the running app will be cached in an HTTP cache. The data will also be stored within the sandbox of the app and are protected against unauthorized access. For this purpose, the AFCache is used in OS|mobileDMS. The AFCache discards all cached data as soon as you exit the app.

Aside from HTTP caching, Favorites documents can be made available for offline use. This might be necessary when documents need to be available for a customer appointment without an existing internet connection. Favorites documents that are available offline will be saved in an unencrypted way in the app's document directory, but are protected by the sandbox of the operating system against unauthorized access. The index data of a document are currently not available in offline mode, if they are not in the AFCache. Downloading and saving a Favorites document to the file system is done automatically as soon as the document is flagged as a favorite in the app.

The document is also removed automatically from the file system of the app when the user removes the document from the Favorites. In addition to the Favorites documents, a list of favorites is available offline and is cached in AFCache and in the file system. This list is updated upon every HTTP retrieval and is saved in the file system in an unencrypted way.

In parallel with the favorites, droptargets are cached in the file system of OS|mobileDMS as an additional data record. Droptargets form the basis for capturing information by using the app. Processes, such as capturing a new contact or a recall are saved in droptargets. From the process information, dynamic forms are created which guide the user through every process step. If the mobile device has a camera, picture can be taken with an available option and will be saved directly in the ECM system. Pictures taken with this option will not be stored in the gallery of the operating system, but, provided that the droptarget was defined respectively, are transferred to the favorites of OS|mobileDMS as new documents.

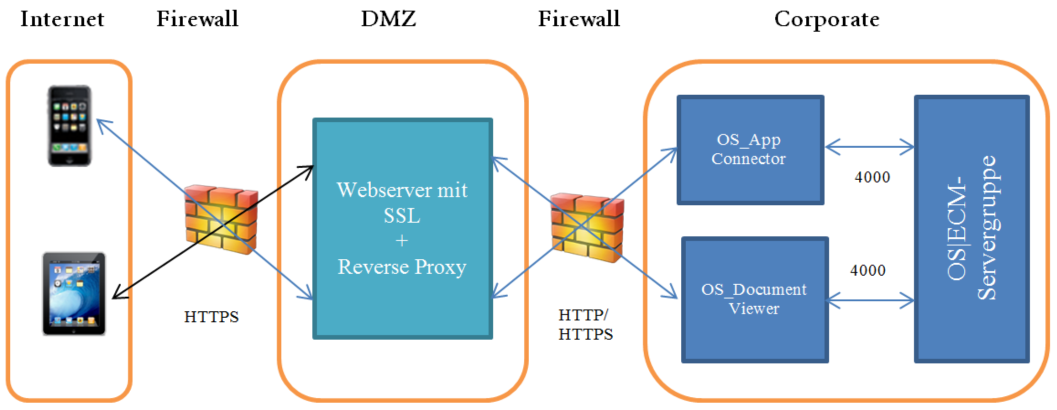
To prevent unauthorized access to data managed in OS|mobileDMS even when the operating system is unlocked, another PIN can be set up in the app. This four-digit PIN will be prompted upon every activation of the app. Furthermore, you can configure in the app settings that after having entered the wrong PIN three times in a row, all saved server profiles in OS|mobileDMS, i.e. the connection data to enaio® servers as well as the user name and password, will be deleted and the demo profile (demo.ecm.mobi) of OPTIMAL SYSTEMS GmbH will be reactivated.

IT Security for enaio® appconnector and enaio® documentviewer

Communication of OS|mobileDMS with enaio® appconnector and enaio® documentviewer is done through HTTP queries. enaio® appconnector transfers all data except documents and their index data to OS|mobileDMS. enaio® documentviewer is started by the app for displaying documents and index data and is embedded into OS|mobileDMS as Web view.

To ensure data security with enaio® appconnector and enaio® documentviewer, the system and network should be structured and configured according to the information security standards (ITG) of the German Federal Office for Information Security (BSI).

The following image depicts how OS|mobileDMS accesses both core services mentioned.



The image shows that OS|mobileDMS which is installed on mobile devices uses an SSL-secured HTTPS connection to access a Web server in the DMZ. This server has a reverse proxy and translates OS|mobileDMS queries into internal HTTPS queries to the enaio® appconnector and enaio® documentviewer core services. Then, the core services request raw data from the enaio® server which is shown as a server group in the image.

If the corporate network consists of several distributed nodes (e.g. computer centers in Stuttgart, Munich, Hamburg etc.), all elements of the corporate network shown in the image should be available for every node. In this case, the Web server with the reverse proxy would translate the query to one specific node depending on the query URL. A similar approach can be followed in a multi-client system. For each client of an enaio® server, a corporate node can be created. The Web server in the DMZ can then translate the query URL to the individual clients and send it to the correct corporate node.

Archiving

Backup

File backup outside of enaio® is one of the most important methods to avoid data losses. File backup is not an administrative activity in the enaio® system. Great backup strategies can usually be found in relevant books and references so only most important backup information will be listed here.

The following should be saved systematically:

- Data

  Files contained in the enaio® database tables must be saved on the enaio® database server. It is strongly recommended to save these files. A defective enaio® system cannot be recovered without continuous backup of the necessary files.

- Document Files

  Document files archived in enaio® are saved in the WORK directory of the application server or the application server group and on archive media. It is strongly recommended to back up the WORK directory and to mirror or copy the integrated storage media. A defective enaio® system cannot be recovered without a backup of the WORK directory and the archive media.

- Program directories:

  - Server

  It is possible to back up program directories. But it is not required. The setup DVD and original license files allow you to newly install the application and file servers. To do so, you must save the license files. In addition to the program directories it is recommended to back up the system registry of the server, as well.

  - Client

  The backup of client installations might be helpful but is not required as you can quickly run network installations at individual workstations. It will be very time intensive to individually back up a great number of clients.

A backup should include all data and document files of the same system status at one point of time. This is to avoid inconsistencies like an index data set without document files. For that reason, run the backup when the system is not productive.

The backup of all data, document files and server program directories allows you to recover an enaio® system by reloading the backup into the database, the WORK directory, the program directories, and the system registry of the application server.

If an enaio® system is not working anymore due to a defective hard disk, document files saved with the last backup will be lost as well. Having installed the database on the same hard disk will result in a loss of the recently backed up index data. A systematic archive with digitally captured paper records should enable you to quickly re-archive lost document files.

Basics

Introduction

One of the main functions of a DMS system is digital archiving of electronic documents. The archiving method must fulfill legal requirements to enable digitalized documents to be recognized as audit-proof representation of a paper document. Amongst others these requirements include unmodified storage of documents on adequate media, such as WORM media (write once read many).

In enaio®, archiving stands for the transfer of digitally captured documents from a hard disk of an application server to archive media, e.g. WORM media. Documents will not be archived until the responsible user explicitly sets the document to 'archivable'. The archiving operation is configured as an automatic action, and then executed as individual batch process by enaio® administrator or enaio® start. The archiving process consists of the document transfer to other storage media and its removal from the WORK area.

If required, you can archive documents in audit-proof manner to external systems that have been integrated. For example, EMC Centera, iTernity and iXOS are available. Documents are then passed to these systems that manage them independently.

Even if the archive media do not fulfill the above mentioned requirements, the archiving process can be executed. Documents will then be filed through the same processes but will not be tamper-proof. It is recommended to regularly outsource specific documents from the WORK area to external media, although the user is not compelled by legal regulations or other motives to archive documents audit-proof. In such a case, data and documents can simply be archived into directories on available hard disks.

The interaction of all relevant factors, such as used procedures, applied hardware and software components, is essential to guarantee that archiving processes fulfill all legal requirements.

It is recommended to have all aspects coming into question be regulated and evidenced by process documentation That is created by the liable operator. If required, OPTIMAL SYSTEMS can provide project support in this matter.

For all compliance storage solutions certified by OPTIMAL SYSTEMS enaio® provides archiving processes that fulfill legal requirements. Correct archiving of documents and physical retention periods are subject to restrictions of the storage system in use. If, for example, a compliance storage system is able to manage retention periods only until 2038, such a restriction cannot be compensated by an archiving software solution like enaio®.

To guarantee simple configuration and secure operation, enaio® provides tools and means for different certified archive storage systems. Nevertheless, keep in mind to follow the configuration steps described in the respective interface manuals.

It is therefore recommended to coordinate, realize, document, and test the planning of retention periods, the selection of an archive storage system and its configuration, the configuration of retention periods as well as necessary archive storage system settings in enaio®, and the correct configuration and execution of archiving processes with our consulting department.

Working Schema with Archived and Not-archived Documents

enaio® is a LAN oriented DMS with a client-server architecture. The following illustration displays the general architecture of enaio®. An application server runs multiple client applications (enaio® client, enaio® editor, enaio® administrator, and enaio® capture) and controls the archiving.

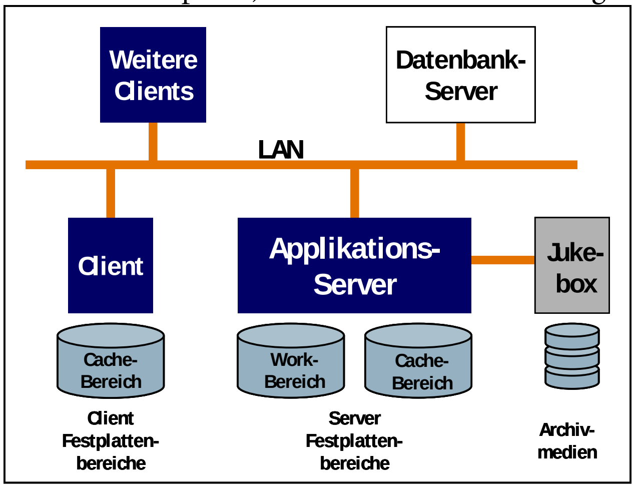


Figure 32: enaio® server-client architecture

In the following, four processes will be described: creating a new document, opening an existing document, archiving a document, and opening an archived document. File operations are displayed only; database processes are not included:

Creating a new document:

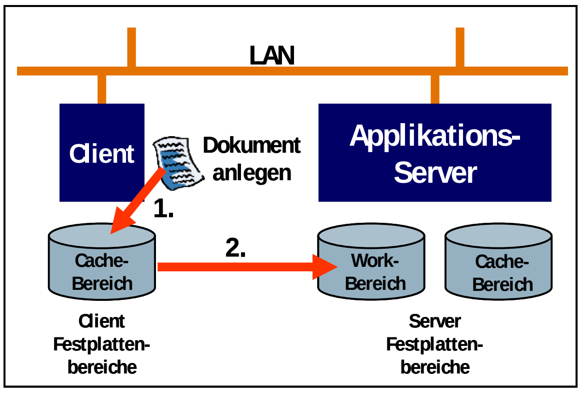


Figure 33: Creating a new document

1. In enaio® client a new document is created. The respective file is saved to the CACHE area of the client.
2. After having closed the document, the file is transferred to the WORK area of the application server and deleted from the client's CACHE area. This process is called 'check in'.

Opening document for editing:

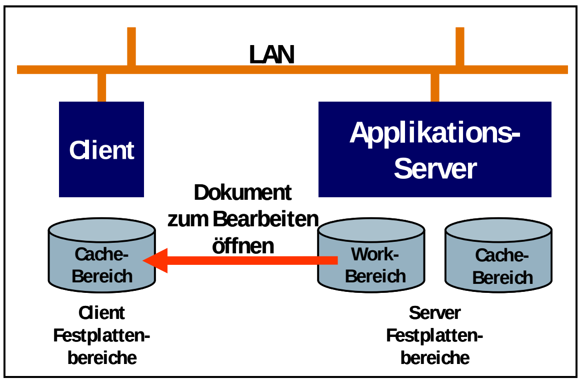

LAN
Client
Applikations-Server
Dokument zum Bearbeiten öffnen
Cache-Bereich
Work-Bereich
Cache-Bereich
Client Festplatten-bereiche
Server Festplatten-bereiche

Figure 34: Opening a document for editing

The respective file is transferred from the WORK area of the application server to the CACHE area of the client. The file is 'checked out' and other users can open a read-only version of the document only.

Archiving a document:

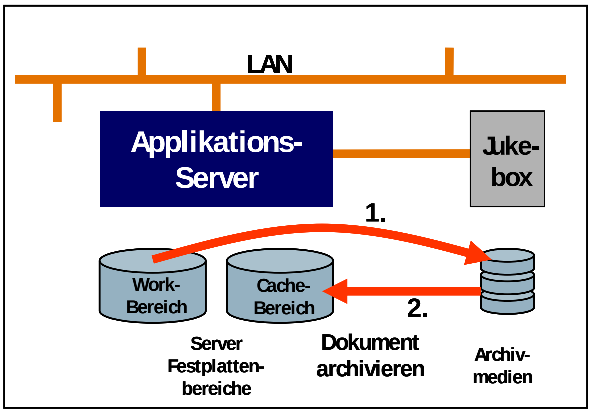


Figure 35: Archiving a document

1. An existing document is copied from the WORK area of the application server into a media and deleted from the WORK area.
2. Afterwards, a file copy is created in the CACHE area of the application server.

Opening archived documents:

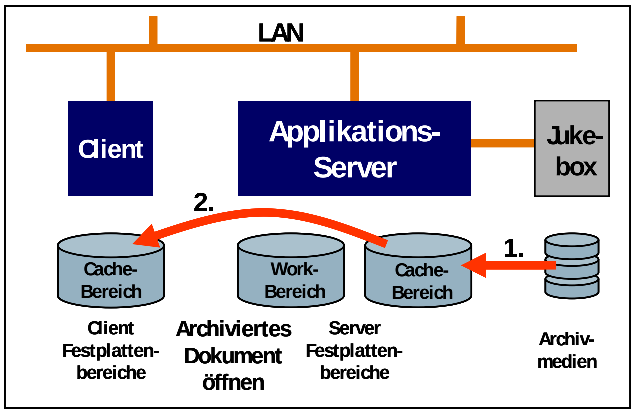


Figure 36: Opening a document

1. If the document is available in the CACHE area of the application server, a read-only version of it is copied into the CACHE area of the client.

2. If there is no file copy available, the document is copied from the archive medium into the CACHE area of the application server and then a read-only version of it is copied into the client's CACHE area.

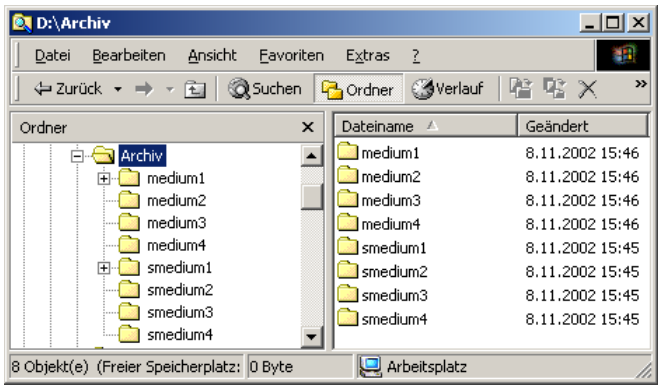
Media, Media Sets and Mirrored Media

Documents are archived in audit-proof manner to WORM media in jukeboxes. Similar to jukeboxes of the entertainment industry, a jukebox for archiving consists of multiple players (drives) and many slots the actual storage media are located in. enaio® supports only jukeboxes addressed by the Pegasus InveStore software and integrated using the file system of the operating system. When setting up a jukebox under InveStore and loading it with storage media, only one drive letter represents the jukebox in the file system regardless of the number of actually charged drives.

The storage media charged in a jukebox appear as directories of this drive. As a consequence, the following terms are commonly used concerning the archiving:

- Media are storage media contained in a jukebox (WORM media) and each side of double-sided storage media is taken as an individual media. Each media is depicted with its formatting name as a separate directory.
- Mirrored media are media duplicating archived information that are assigned to original media for enhanced data security. A mirrored media is an exact copy of an original media. With mirrored media, there are two backup copies containing archived documents.
- Media sets include more than one directory (media). They structure the archiving and simplify the archive management. Media sets contain any media to be specified by the user involving mirrored media for reasons of data security.

The following figure displays an example of the structure of an archive with media (`medium1` to `medium4`) and mirrored media (`smedium1` to `smedium4`):



Dateiname	Geändert
medium1	8.11.2002 15:46
medium2	8.11.2002 15:46
medium3	8.11.2002 15:46
medium4	8.11.2002 15:46
smedium1	8.11.2002 15:45
smedium2	8.11.2002 15:45
smedium3	8.11.2002 15:45
smedium4	8.11.2002 15:45

8 Objekt(e) (Freier Speicherplatz: 0 Byte Arbeitsplatz

Figure 37: Example of an archive structure with media and mirrored media

The configuration tools offered by enaio® for setting up the archiving process allows you to create any media set. The following chapter will describe the creation of media sets, media, and mirrored media.

The archiving process is set up in enaio® enterprise-manager and its description can be found in the documentation about the automatic action 'Archiving'.

Archiving Process and Access to Archived Documents

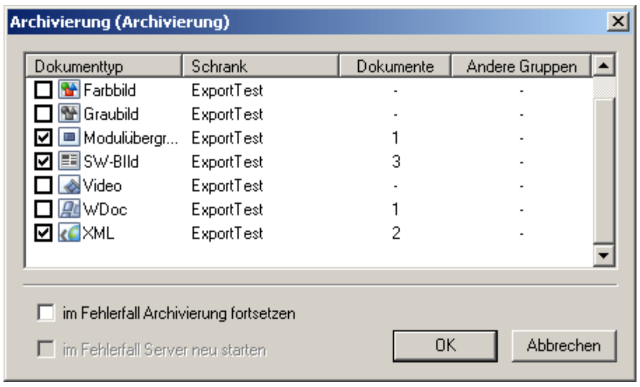
Archiving concept

The archiving concept includes write-once archive media only (like write-once CDs in multi-sessions). Files can neither be deleted nor opened for editing. Before archiving it is to be ensured that there is not only space on the storage media available for the document itself but also for all information concerning the document. For that reason, the algorithm of data storage is quite complicated. enaio® offers two methods for storage space determination on archive media if this media is in a jukebox and can be addressed by Pegasus software. In the registry you can enter `MediaSizeMethod` under `HKLM\SOFTWARE\OPTIMAL SYSTEMS\'AppServer'\Schemata\4.0\Archive`. This is useful for jukeboxes that can be addressed with Pegasus InveStore and can have three values:

0	Identification with the file system (default)
1	Identification with Pegasus-Software (contents of '!FSFREE.###')
2	Identification with Pegasus-Software (size of 'FSFREE__.###')

The method using the file system is faster whereas the identification is more precise when applying the Pegasus software. With the first method you will not risk to lose data but it will possibly result in less data stored on a media due to a lack of space. For media on hard disks this parameter is irrelevant.

When configuring the automatic action 'Archiving', a configuration dialog will display all document types connected with media sets in the database table `medrel` (see Figure 42). As well, the number of documents flagged as archivable for each object type is displayed to the user. During configuration, only archivable document types can be selected.


Archivierung (Archivierung)

	Dokumenttyp	Schrank	Dokumente	Andere Gruppen
☐	Farbbild	ExportTest	-	-
☐	Graubild	ExportTest	-	-
☑	Modulübergr...	ExportTest	1	-
☑	SW-BIld	ExportTest	3	-
☐	Video	ExportTest	-	-
☐	WDoc	ExportTest	1	-
☑	XML	ExportTest	2	-

☐ im Fehlerfall Archivierung fortsetzen

☐ im Fehlerfall Server neu starten

Figure 38: Selection of document types to be archived

For each media, the media size, the cluster size, and the minimum storage capacity to be kept free are defined. Together with these values and the already occupied storage space, the storage space still free for archiving can be determined.

Afterwards, the number of files of an object type to be archived, its size, and the size of recovery files `stamm#.xxx`, `regis#.xxx`, `object#.xxx` containing index information is determined. If there is enough space for all files, they will be copied onto the media. Only complete documents are archived on media. If a document consists of more than one files (pages), all document files (pages) must be saved on the media. Otherwise, this document will be archived on the media available next. The archiving is an incremental process. According to the available storage space, all documents of the current document type are copied onto the archive medium. The following file structure will be created:

```
Path\\Archive_name\\Media_name\\Main_type\\Sub_type\\Document_type\\DocID
(mod 0x100)\\DocID.Page_number
```

or

`Path\\Archive_name\\Media_name\\Main_type\\Sub_type\\Document_type\\DocID (mod 0x100)\\DocID.ext`

The sub-directory 'Document type' can only have two values:

- '02' for the slide of a document and
- '03' for the actual document.

For W-, M-, E-, and XML documents, 'ext' must be replaced with the correct document extension.

On all media used for archiving, the archiving program creates the file `as_label.dat` the media name is entered in, in this case it is `MEDIUM1`.

Furthermore, the directory `SYS` is created on the media recovery files with index data and object definitions valid at the archiving time are filed in at each archiving process. The recovery files are ASCII text files in which the indexing of all folders, registers, and documents of each archived document are entered. Hence, the files `stamm#.xxx`, `regis#.xxx`, `object#.xxx`, and `sdrel.xxx` are created (The three-digit number `xxx` depends on the incremental status.). These files have a mini header as follows:

- First row: version information
- Second row: existing fields (fields are separated through tabs)
- Third row through nth row: data (fields are separated through tabs)

Example for a file including recovery data (here for two folders in cabinet #1)

```
stamm1,Dokumente, optimal_AS Version 3.x * optimal systems GmbH #id,Ordner
id              feld1
19              documents in the main group
45              documents in the subsidiary 1
```

File `stamm1.006`

Keep in mind that cabinet and register data must not be redundant when entering the index data.

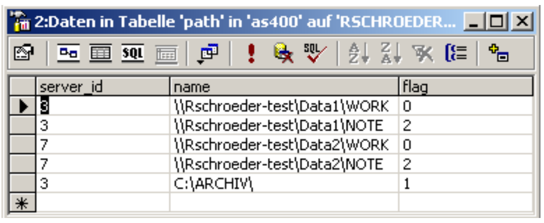
The directory `ARCHIVE` is created on the same level as the directory `WORK` and contains a sub-directory for each media named according to the respective media. Into the sub-directory `_SYS` all recovery files of the most recent archiving process are filed. Furthermore, the file `recover.ini` is created here in the media directory with the following syntax that contains the number of the most recent archiving process.

```
[SYS]
LNR=6
```

File `recover.ini` from the directory `ARCHIVE`

The following database table provide information on the archiving process

1. Database table `path`:



2:Daten in Tabelle 'path' in 'as400' auf 'RSCHROEDER...

server_id	name	flag
3	\\Rschroeder-test\Data1\WORK	0
3	\\Rschroeder-test\Data1\NOTE	2
7	\\Rschroeder-test\Data2\WORK	0
7	\\Rschroeder-test\Data2\NOTE	2
3	C:\ARCHIV\	1

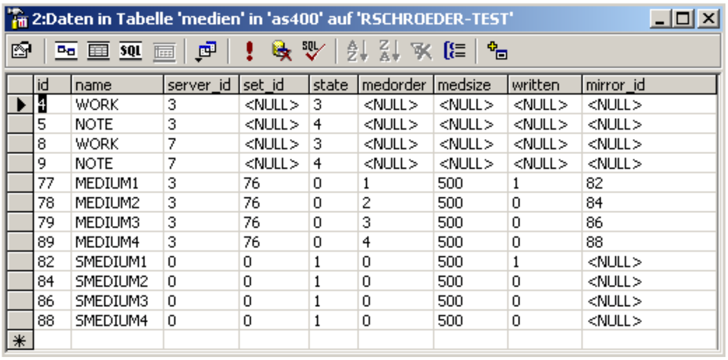
Figure 39: Table `path` for paths known in the system

Into the table `path` paths defined for each server group are entered. The field `server_id` means here `servergroup_id`. The flags have the following meaning:

- 0 – path to the server WORK
- 1 – archive path

- 2 – path to notes

2. Database table `media`:



2:Daten in Tabelle 'medien' in 'as400' auf 'RSCHROEDER-TEST'

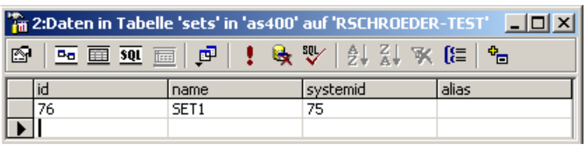
id	name	server_id	set_id	state	medorder	medsize	written	mirror_id
4	WORK	3	<NULL>	3	<NULL>	<NULL>	<NULL>	<NULL>
5	NOTE	3	<NULL>	4	<NULL>	<NULL>	<NULL>	<NULL>
8	WORK	7	<NULL>	3	<NULL>	<NULL>	<NULL>	<NULL>
9	NOTE	7	<NULL>	4	<NULL>	<NULL>	<NULL>	<NULL>
77	MEDIUM1	3	76	0	1	500	1	82
78	MEDIUM2	3	76	0	2	500	0	84
79	MEDIUM3	3	76	0	3	500	0	86
89	MEDIUM4	3	76	0	4	500	0	88
82	SMEDIUM1	0	0	1	0	500	1	<NULL>
84	SMEDIUM2	0	0	1	0	500	0	<NULL>
86	SMEDIUM3	0	0	1	0	500	0	<NULL>
88	SMEDIUM4	0	0	1	0	500	0	<NULL>

Figure 40: Database table `media` for media assignment to media sets

To each media in the table a unique ID is assigned to from the table `osnextindex`. The table `media` performs more than one function:

- the field `state` informs about the media type
  0 free
  1 locked
  2 without meaning
  3 WORK area
  4 NOTE area
- `server_id` does again mean `servergroup_id`,
- with the field `set_id` registered media are assigned to chosen media sets,
- with the field `mirror_id` mirrored media are assigned to the archive media.

3. Database table `sets`:



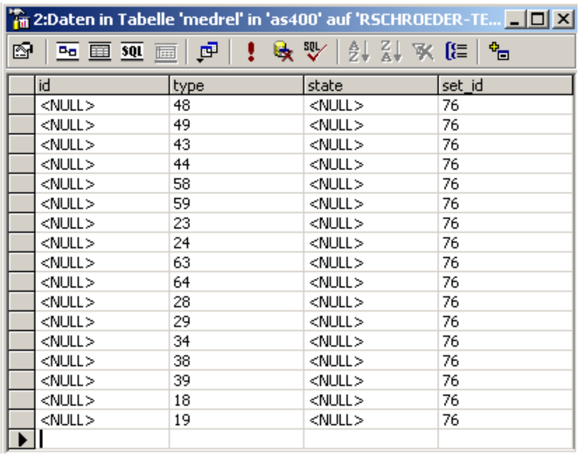
2:Daten in Tabelle 'sets' in 'as400' auf 'RSCHROEDER-TEST'

id	name	systemid	alias
76	SET1	75	

Figure 41: Database table `sets` for media sets

Into this table, all media sets, a system-wide but unique ID, and a media set name are entered. The ID of a set is entered into the table `media` as `set_id` and allows you to assign an archive medium to a media set.

4. Database table `medrel`:



2:Daten in Tabelle 'medrel' in 'as400' auf 'RSCHROEDER-TE...

id	type	state	set_id
<NULL>	48	<NULL>	76
<NULL>	49	<NULL>	76
<NULL>	43	<NULL>	76
<NULL>	44	<NULL>	76
<NULL>	58	<NULL>	76
<NULL>	59	<NULL>	76
<NULL>	23	<NULL>	76
<NULL>	24	<NULL>	76
<NULL>	63	<NULL>	76
<NULL>	64	<NULL>	76
<NULL>	28	<NULL>	76
<NULL>	29	<NULL>	76
<NULL>	34	<NULL>	76
<NULL>	38	<NULL>	76
<NULL>	39	<NULL>	76
<NULL>	18	<NULL>	76
<NULL>	19	<NULL>	76

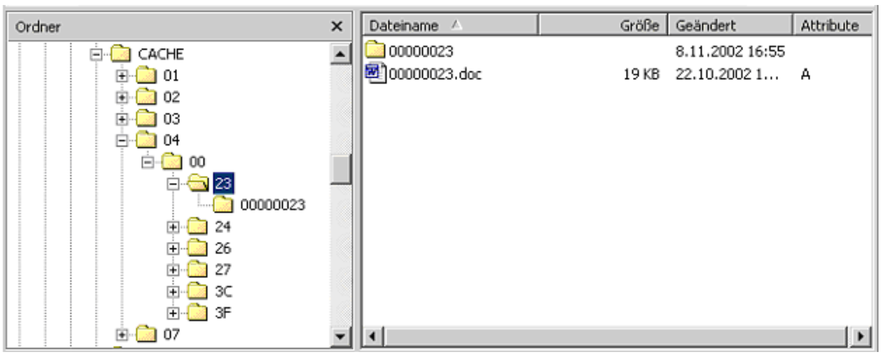
Figure 42: Database table `medrel` for assignment of media sets to document types

If object types from the object definition have been assigned to a media set at media set configuration, these object types are entered here. The field `type` contains the usual main and sub types in form of 2 byte variables due to specific conversion.

The automatic action 'Archiving' (`axacarch.dll`) that is integrated, configured, and launched in enaio® administrator, starts the archiving process. In addition, with enaio® start you can define the archiving process to be started periodically or at a certain time. More information on the automatic action 'Archiving' can be found in the corresponding handbook about automatic actions.

During each archiving process, object definition data are automatically written to the archive media for reasons of auditing acceptability. If only a few documents but many object definition data are archived, it may be useful to not write the object definition to the archive media and to ensure auditing acceptability through method documentations. This is why the archiving of object definition data can be turned off in enaio® enterprise-manager.

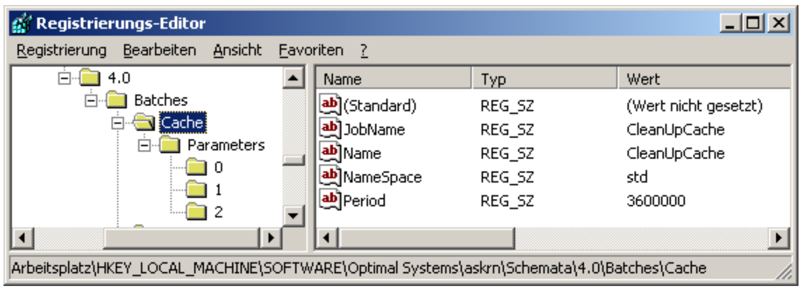
The archiving process is successful after the respective document has been transferred from the WORK area to the CACHE area (see Figure 43) of the server, i.e. it has been created in the CACHE and deleted from the WORK. The CACHE directory has the same structure as the file system in the WORK area so the same algorithm for document access can be employed. The CACHE area allows you to faster access data as after an archiving process files cannot be modified anymore. If a client requests an archived document from the server, the document is searched in the CACHE area first. If the document is found in the CACHE, it will be made available for the client. Hard disks are accessed faster than archive media; thus, the document can be provided faster.



Dateiname	Größe	Geändert	Attribute
00000023		8.11.2002 16:55	
00000023.doc	19 KB	22.10.2002 1...	A

Figure 43: Example for a CACHE directory of a server after an archiving process

The application server offers periodic actions for the administration of CACHE area parameters to avoid an obsolete or oversized document inventory. In the registry of a particular server instance, you can define the following in the subdirectory BATCHES:

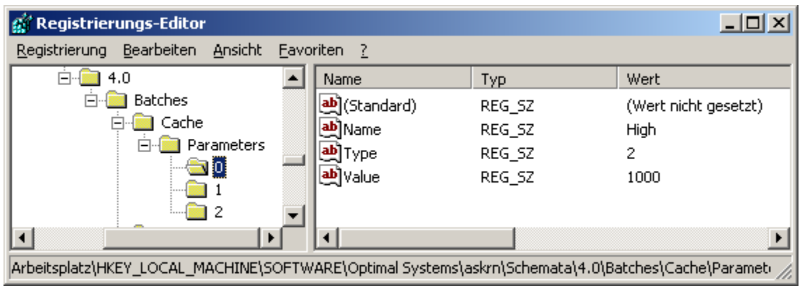


Name	Typ	Wert
(Standard)	REG_SZ	(Wert nicht gesetzt)
JobName	REG_SZ	CleanUpCache
Name	REG_SZ	CleanUpCache
NameSpace	REG_SZ	std
Period	REG_SZ	3600000

Arbeitsplatz\HKEY_LOCAL_MACHINE\SOFTWARE\Optimal Systems\askrn\Schemata\4.0\Batches\Cache

Figure 44: CACHE area parameters

The server job 'CleanUpCache' of the namespace (executor) 'std' runs every 60 minutes (3600000 ms comply with 60 minutes) to check the CACHE. The specified parameter defines the check:

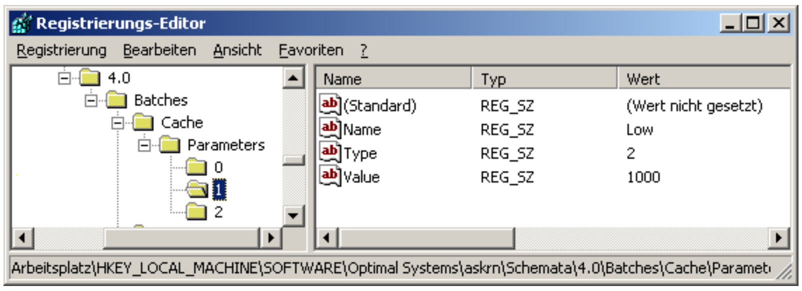


Name	Typ	Wert
(Standard)	REG_SZ	(Wert nicht gesetzt)
Name	REG_SZ	High
Type	REG_SZ	2
Value	REG_SZ	1000

Arbeitsplatz\HKEY_LOCAL_MACHINE\SOFTWARE\Optimal Systems\askrn\Schemata\4.0\Batches\Cache\Paramet

Figure 45: Value 'High' set for fill level of the CACHE

The value 'High' indicates the maximum size (in MB), whereas type=2 means the value is an integer.

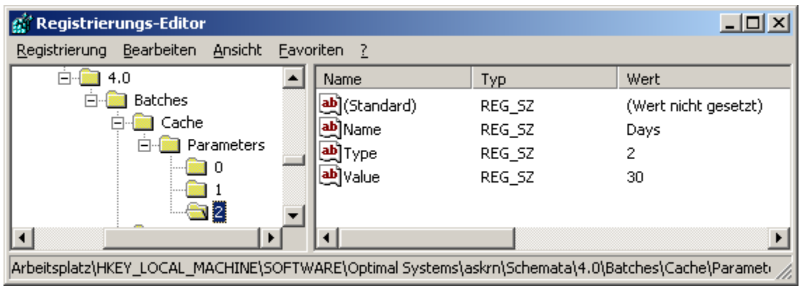


Name	Typ	Wert
(Standard)	REG_SZ	(Wert nicht gesetzt)
Name	REG_SZ	Low
Type	REG_SZ	2
Value	REG_SZ	1000

Arbeitsplatz\HKEY_LOCAL_MACHINE\SOFTWARE\Optimal Systems\askrn\Schemata\4.0\Batches\Cache\Paramet

Figure 46: Value 'Low' set for fill level of the CACHE

The value 'Low' indicates the minimum size (in MB), whereas type=2 means the value is an integer.



Name	Typ	Wert
(Standard)	REG_SZ	(Wert nicht gesetzt)
Name	REG_SZ	Days
Type	REG_SZ	2
Value	REG_SZ	30

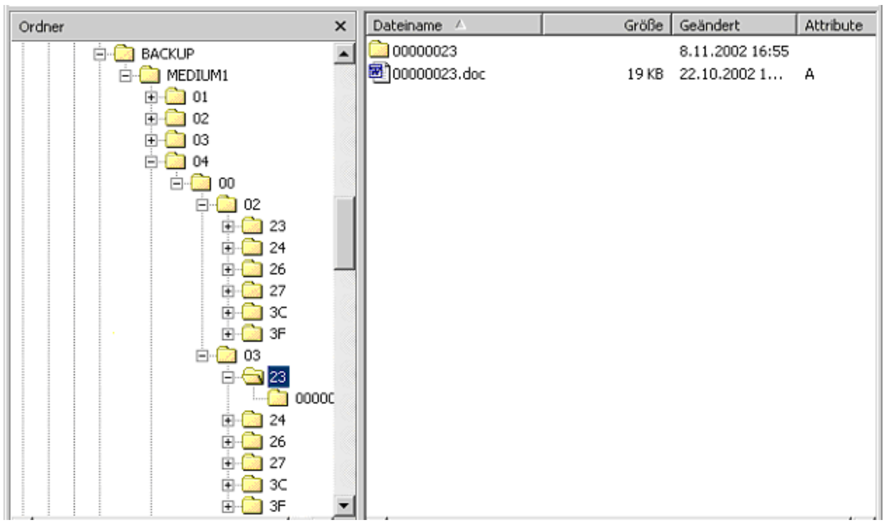
Arbeitsplatz\HKEY_LOCAL_MACHINE\SOFTWARE\Optimal Systems\askrn\Schemata\4.0\Batches\Cache\Paramet

Figure 47: Holding time of a document in the CACHE

The value 'Days' indicates the time period (in days) after that all files will be removed being longer than the specified number of days in the CACHE without being read. However, they will not be removed until the top value has been reached. Files are deleted until either the minimum value has been reached or there are no more files being older than specified. The file age has always priority.

The following algorithm defines the CACHE cleaning. It is periodically checked whether or not the CACHE sizes is exceeded. If yes, CACHE directories are consecutively searched for files meeting the condition. If such a file is found, it is deleted. In case the CACHE now has reached the indicated minimum size, the cleaning process is cancelled, and if not, the file search continues. If no file meets the deletion condition, no file will be removed and the CACHE size exceeds temporarily. That is why you should define an adequate CACHE size and keep enough free space on the hard disk in order to avoid file system errors due to the last described scenario.

In enaio® administrator you can configure in the archiving configuration that a copy is created of each archived file to be filed into the BACKUP directory in order to improve data security during the archiving process (see Figure 48). This directory has the same structure as the file system of the archive medium.



Dateiname	Größe	Geändert	Attribute
00000023		8.11.2002 16:55	
00000023.doc	19 KB	22.10.2002 1...	A

Figure 48: Backup directory of an archiving for enhanced data security

Logging

Each archiving process is logged in a file all archived document types and error descriptions are entered into. This file has the name `osYYMMDD.rep` (YY- two-digit year, MM- two-digit month, DD- two-digit day). In case there are two archiving processes at the same day, the log will be updated in the file. The log contains the following information:

- Date and time of the report
- which document type was supposed to be archived,
- when the action was started,
- in which server group the documents were found,
- by which server of the group archiving was performed,
- how many documents were found as to be archived,
- if register information was found,
- the found media and mirrored media,
- errors,
- calculation of free storage space on the media,
- result, how many documents were archived and
- a summary of the results.

This information is indicated for each archived document type. The report is created for each archiving process. It is furthermore possible to get more detailed information on the archiving process. To achieve this, the option 'Confirmed archiving' must be enabled in enaio® enterprise-manager. As a result, the files `osDDMMYY.ext` are created on the same level as the archive report. 'ext' is a file extension and follows the schema:

- character 1 = main document type, e.g. 1 for X-documents
- character 2,3 = sub document type, e.g. 00 for the first appearance of X-documents in the object definition

The following document types are available:

- osDDMMYY.100 to osDDMMYY.1FF for X-documents
- osDDMMYY.200 to osDDMMYY.2FF for D-documents
- osDDMMYY.300 to osDDMMYY.3FF for P-documents

- osDDMMYY.400 to osDDMMYY.4FF for W-documents
- osTTMMJJ.500 to osTTMMJJ.5FF for video documents
- osDDMMYY.600 to osDDMMYY.6FF for e-mail documents
- osDDMMYY.700 to osDDMMYY.7FF for XML documents

Detailed log files belonging to the above described archiving process are listed below:

Folder	Label	Media	Path
Documents in the main group	X-Dok 1 at HG	MEDIUM1	01\00\03\1D\0000001d.000
Documents in the main group	X-Dok 2 at HG	MEDIUM1	01\00\03\1E\0000001e.000
Documents in the main group	X-Dok 3 at HG	MEDIUM1	01\00\03\1F\0000001f.000

File `os081102.100` *for the archiving of X-documents*

Folder	Label	Media	Path
Documents in the subsidiary 1	D-Dok 1 at NS	MEDIUM1	02\00\03\33\00000033.000
Documents in the subsidiary 1	D-Dok 2 at NS	MEDIUM1	02\00\03\34\00000034.000
Documents in the subsidiary 1	D-Dok 3 at NS	MEDIUM1	02\00\03\35\00000035.000

File `os081102.200` *for the archiving of D-documents*

Folder	Label	Media	Path
Documents in the main group	P-Dok 1 at HG	MEDIUM1	03\00\03\20\00000020.000
Documents in the main group	P-Dok 2 at HG	MEDIUM1	03\00\03\21\00000021.000
Documents in the main group	P-Dok 3 at HG	MEDIUM1	03\00\03\22\00000022.000

File `os081102.300` *for the archiving of P-documents*

Folder	Label	Media	Path
Documents in the main group	Brief 1at HG	MEDIUM1	04\00\03\23\00000023.doc
Documents in the main group	Brief 2 at HG	MEDIUM1	04\00\03\24\00000024.doc
Documents in the main group	Info 1 at HG	MEDIUM1	04\00\03\26\00000026.doc
Documents in the main group	Info 2	MEDIUM1	04\00\03\27\00000027.doc
Documents in the subsidiary 1	Brief 1 at NS	MEDIUM1	04\00\03\3C\0000003C.doc
Documents in the subsidiary 1	Info 2 at NS	MEDIUM1	04\00\03\3F\0000003F.doc

File `os081102.400` *for the archiving of W-documents*

Folder	Label	Media	Path
Documents in the main group	simple 1 at HG	MEDIUM1	07\00\03\28\00000028.xml
Documents in the main group	invoice 1 at HG	MEDIUM1	07\00\03\29\00000029.xml
Documents in the main group	mbox 1 at HG	MEDIUM1	07\00\03\2A\0000002a.xml
Documents in the main group	mbox 2 at HG	MEDIUM1	07\00\03\2B\0000002b.xml
Documents in the main group	test 1 at HG	MEDIUM1	07\00\03\2C\0000002c.xml
Documents in the subsidiary 1	simple 1 at NS	MEDIUM1	07\00\03\40\00000040.xml

File `os081102.700` *for the archiving of XML documents*

Into each file, the indexing of folders, registers, and documents (if available), media name, and full file name of the archived document is entered.

Working with Archived Documents

The following is an example scenario. A user is searching a specific document and wants to open it in read-only mode. To do so, he must use a search form to start a query that corresponds to the searched document type:

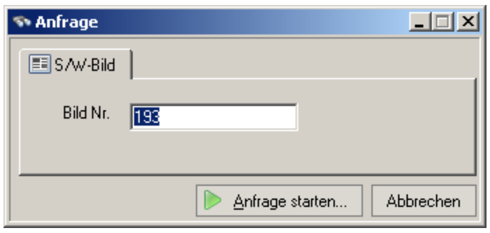


*Figure 49: Query form for D-document with image no. 193**

With the selected search form, enaio® client is informed about the main and sub document type as well as the index field the search term should be searched in.

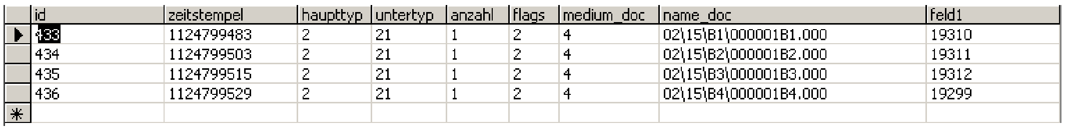
In this example these are:

- main type: 2 (D-document)
- sub type: 0 (as information out of the object definition)
- search term: 193* (term extension has been enabled on the tab 'Auto' before)

The search is performed using the following SQL statement:

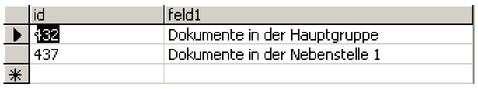
```
SELECT d.feld1, d.id, d.zeitstempel, d.version, d.links, d.flags,
 d.lockuser, d.haupttyp, d.anzahl, d.medium_doc
FROM object2 d, stamm1 o, sdrel x
WHERE x.stamm_id = o.id and x.object_id = d.id and
 ( {fn UCASE(d.feld1)} like '193%')
```

The here integrated database tables `object2`, `stamm1`, and `sdrel` contain the following:



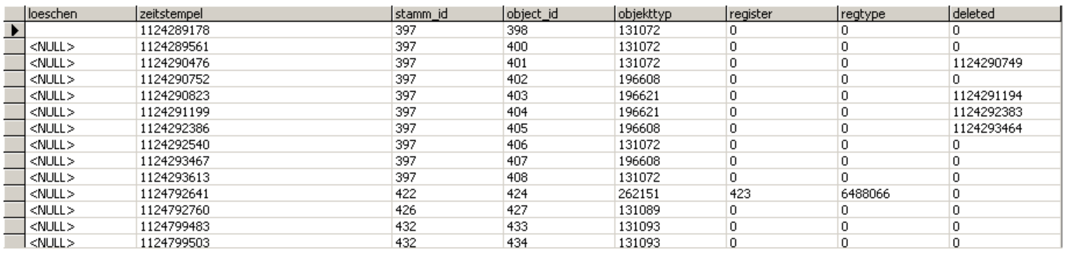
id	zeitstempel	haupttyp	untertyp	anzahl	flags	medium_doc	name_doc	feld1
433	1124799483	2	21	1	2	4	02\15\B1\000001B1.000	19310
434	1124799503	2	21	1	2	4	02\15\B2\000001B2.000	19311
435	1124799515	2	21	1	2	4	02\15\B3\000001B3.000	19312
436	1124799529	2	21	1	2	4	02\15\B4\000001B4.000	19299

Figure 50: Database table `object2` with six D-documents



id	feld1
432	Dokumente in der Hauptgruppe
437	Dokumente in der Nebenstelle 1

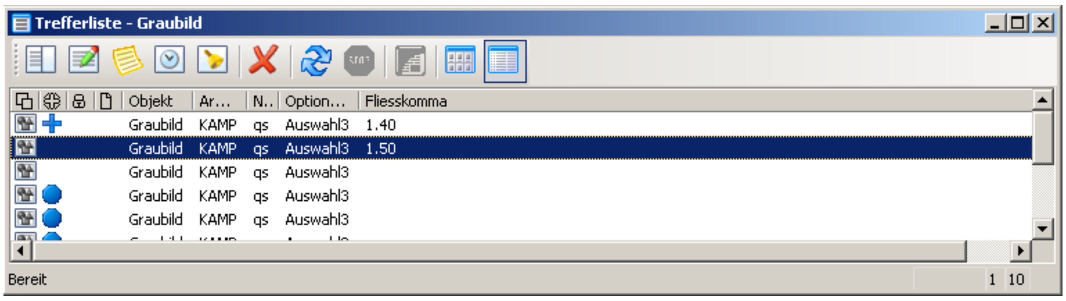
Figure 51: Database table `stamm1` with two folders



loeschen	zeitstempel	stamm_id	object_id	objekttyp	register	regtype	deleted
	1124289178	397	398	131072	0	0	0
<NULL>	1124289561	397	400	131072	0	0	0
<NULL>	1124290476	397	401	131072	0	0	1124290749
<NULL>	1124290752	397	402	196608	0	0	0
<NULL>	1124290823	397	403	196621	0	0	1124291194
<NULL>	1124291199	397	404	196621	0	0	1124292383
<NULL>	1124292386	397	405	196608	0	0	1124293464
<NULL>	1124292540	397	406	131072	0	0	0
<NULL>	1124293467	397	407	196608	0	0	0
<NULL>	1124293613	397	408	131072	0	0	0
<NULL>	1124792641	422	424	262151	423	6488066	0
<NULL>	1124792760	426	427	131089	0	0	0
<NULL>	1124799483	432	433	131093	0	0	0
<NULL>	1124799503	432	434	131093	0	0	0

Figure 52: Database table `sdrel` for document-folder-register relations

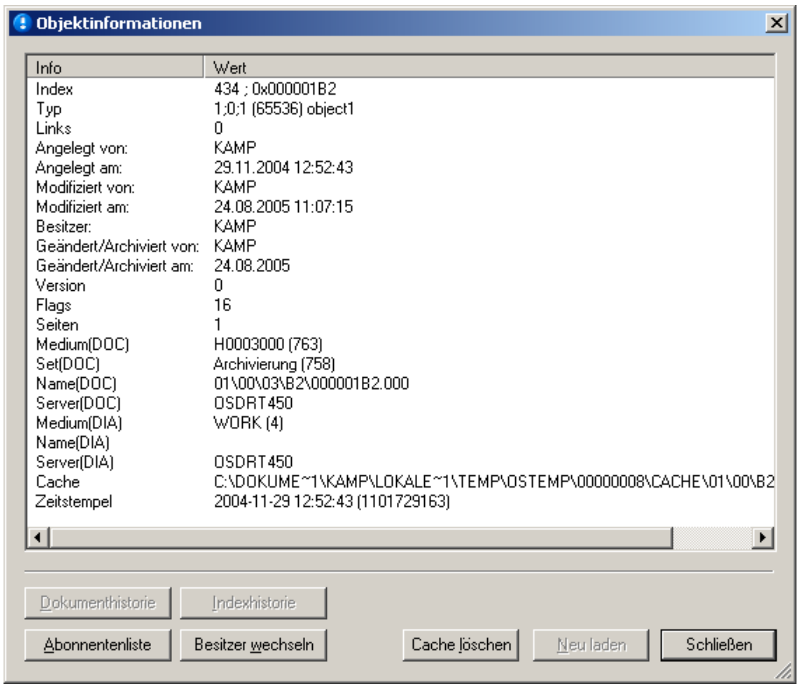
The following hit list is the result of the executed query:



Objekt	Ar...	N..	Option...	Fliesskomma
Graubild	KAMP	qs	Auswahl3	1.40
Graubild	KAMP	qs	Auswahl3	1.50
Graubild	KAMP	qs	Auswahl3	
Graubild	KAMP	qs	Auswahl3	
Graubild	KAMP	qs	Auswahl3	

Figure 53: Hit list of a search for D-documents

As you would expect, numerous documents have been found (see Figure 50), among which three are flagged as archived (`object2,flag=0`) and three as archivable (`object2.flag=1`). Use the F8 key to view the following information on the selected, archived document:



Objektinformationen

Info	Wert
Index	434 ; 0x000001B2
Typ	1;0;1 (65536) object1
Links	0
Angelegt von:	KAMP
Angelegt am:	29.11.2004 12:52:43
Modifiziert von:	KAMP
Modifiziert am:	24.08.2005 11:07:15
Besitzer:	KAMP
Geändert/Archiviert von:	KAMP
Geändert/Archiviert am:	24.08.2005
Version	0
Flags	16
Seiten	1
Medium(DOC)	H0003000 (763)
Set(DOC)	Archivierung (758)
Name(DOC)	01\00\03\B2\000001B2.000
Server(DOC)	OSDRT450
Medium(DIA)	WORK (4)
Name(DIA)	
Server(DIA)	OSDRT450
Cache	C:\DOKUME~1\KAMP\LOKALE~1\TEMP\OSTEMP\00000008\CACHE\01\00\B2
Zeitstempel	2004-11-29 12:52:43 (1101729163)

Figure 54: Information on the document with id=434 (0x1B2)

Now, request this document from the application server. The document may not be in the CACHE area but must be transferred from the archive medium. To do so, enter the full file name (archive path, medium and name in the file system).

The following statement allows you to determine whether or not the document is entered properly into the database tables.

```
SELECT d.feld1, d.id, d.zeitstempel, d.version, d.links, d.flags,
 d.lockuser, d.haupttyp, d.anzahl, d.medium_doc
FROM object2 d, stamm1 o, sdrel x
WHERE d.id = 433 and x.object_id=d.id and x.stamm_id=o.id
```

Afterwards, the document's indexing and system properties are loaded.

Indexing

```
select feld1 from object2 where id = 433
```

System properties

```
SELECT zeitstempel, haupttyp, untertyp, anzahl, flags, medium_doc,
 medium_dia, name_doc, name_dia, angelegt, anleger,
 archiviert, archivar, lockuser, foreignid, systemid
FROM object2
WHERE id=433
```

Therewith, following information are available for further SQL queries and for document display:

- `flag=0` means 'archived'
- `medium_doc` is the ID of the archive medium (4)
- `medium_name` is the file system path of the archive medium (`02\15\B1\000001B1.000`)

Especially, `object2.medium_name='02\00\03\33\00000033.000'` is already part of the full file name of the document to be displayed.

Now, the name of the archive medium can be determined.

```
SELECT name, mirror_id, set_id, server_id
FROM medien
WHERE id=4
```

The values for `name='Medium1'` (medien.name='Medium1'), `mirror_id`=82, `set_id`=76, `server_id=3` are therewith known and the mirrored media name can be determined.

```
SELECT m2.name
FROM medien m1, medien m2
WHERE m1.name ='MEDIUM1' AND m1.mirror_id = m2.id
```

Finally, the archive path for the server group `server_id=3` must be determined.

```
SELECT name
FROM path
where flag=1 and server_id=3
```

The result is 'path.name= C:\Archiv'. Now, all file location elements are known and the full file name can be created:

`path.name\medien.name\object2.medium_name`

For the here described example the following file name forms up:

`C:\Archiv\MEDIUM1\02\15\B1\000001B1.000`

External Archiving Systems

It is also possible to integrate third-party archiving systems. For example, EMC Centera, iTernity, iXOS or Tivoli Storage.

This is done by integrating a virtual archive, setting up a media set and assigning the desired document types to the media set.

In the manner of archiving with enaio® server, the archiving process itself is carried out as an automatic action.

The mentioned third-party archiving systems usually support retention times and allow deleting specific documents.

Both aspects may be especially important for data protection reasons.

Detailed information about third-party archiving systems can be found in the 'Storage Guide'.

Administration

Overview

In this chapter, operations for the administration of an enaio® system are described. Administration comprises a great number of different operations so at first, the differences between administrative operations in and around the enaio® system and then the differences between administrative and productive operations in an enaio® system are clarified. At last, several administrative operations within an enaio® system are defined and summarized.

Distinction: Administrative Operations outside of the enaio® System

It is important to distinguish between administrative operations within an enaio® system and general, administrative operations concerning the computer environment of the enaio® system. With the latter you can also prepare and enable the operation of enaio® but they will not be described in this System Handbook.

That includes the following operations:

- Hardware installation and setup:
  - Network
  - PCs
  - Jukebox
  - Scanner
  - Backup hardware
  - Other hardware (e.g. OCR dongles)
- Software installation and setup:
  - Operating system
  - Network, domain administration
  - Database
  - E-mail
  - Drivers
  - Office
  - System backup
  - DB backup and maintenance
- Troubleshooting outside of enaio®:
  - Hardware error
  - Operating system error
  - Database error

- Other software errors, e.g. in MS Office, driver errors

All these operations are not described in this handbook.

Distinction: Productive and Administrative Operations within enaio®

Within enaio® it is important to distinguish between administrative and productive operations. Productive operations are performed in enaio® client, enaio® capture, and in other client applications, including, for example, the archive printer or transfer macros in MS Office products. In general, enaio® client and enaio® webclient offer productive operations to all users.

Operations are productive if enaio® users execute one of the following actions in enaio® client, enaio® capture, or in another client application:

- DMS objects are being queried.
- DMS objects are being created.
- DMS objects are being deleted.
- The indexing of the DMS objects is checked.
- The indexing of the DMS objects is changed.
- Document files of the DMS objects are checked.
- Document files of the DMS objects are validated.
- Document files are being deleted.

Apart from these productive operations, users do also configure enaio® client, enaio® capture, and other client applications, e.g. setting up the graphic interface, selecting fields to be displayed, query behavior etc. All these (administrative system) operations are neither described in this handbook.

Operations are administrative if they are executed from specifically authorized persons in order to prepare and configure productive operations in enaio®, and to maintain and extend system functions. These types of administrative operations are called preparation, maintenance and extension of the system.

A special class of administrative operations are automatic import processes, e.g. the dBase-III-import or the HL7 server, both being configured and executed by administrators. These system components can execute nearly all actions above defined as productive operations. However, they are not executed in one of the above mentioned client applications but within an administrative tool and, after configuration, without user intervention.

enaio® Administration

According to their function for the enaio® system, there are different administrative operations: those for preparing the productive application of enaio® and those for system maintenance. Along with it, there are operations offering functions being beyond the scope of productive operations. These are for example: the communication with other systems and the configuration and execution of data imports. Altogether, administrative operations can be separated in:

- Preparing the productive application
- System maintenance
- Extended operations

In the following, these three operation areas are summarized.

Preparing the productive application of enaio®

A number of administrative operations is required to create an object in enaio® client at all. As least the following is required:

- Installation of an enaio® system (see chapter 'Installation' offered in this handbook)
- Definition of the objects as follows in enaio® editor:
- a cabinet
- a document type, e.g. D-document
- Configuration of the security system in enaio® administrator:
- creation of a user group
- creation of a user for this user group
- assignment of created objects to the user group
- allocation of access permissions for this object to the user group
- Configuration of the license system in enaio® enterprise-manager:
- adding a new station
- assignment of necessary license modules to the new station

After these operations and the necessary restart of the application server, a user is capable to start enaio® client, to create folders in a cabinet, and to create black-and-white documents in these folders. A user can apply a scanner for document capture in case there is an available scanner configured for the workstation the user works on.

Therewith, just a small part of available system properties has been configured. More configuration is required if full system performance should be used.

- Definition of further objects:
  - cabinets
  - registers
  - document types
- Configuration of the security system in enaio® administrator:
  - creation of more than one user group
  - creation of more users and their assignment to user groups
  - differentiated assignment of created object type to user groups
  - differentiated allocation of access permissions for objects to user groups
- Setup of W-modules for W-document types:
  - preparation of document templates (no enaio®-specific administrative operation)
  - setup of Word and Excel template for data import
- Configuration of the license system in enaio® enterprise-manager:
  - adding all necessary stations
  - assignment of necessary license modules to the new stations
- Setup of D-modules to enable archive printer application
- configuration of events and other scripting (os_events, AddOn, enaio® capture scripts)
- manual entry of configuration files (as.cfg etc.) and registry
- enaio® capture administration (see chapter 'enaio® capture')

- workflow setup (see System handbook 'Workflow Management System')

System maintenance

The second group of operations serves for maintenance and optimization of set up system functions. The following is included:

- logging setup
- repair installations
- troubleshooting
- setup and execution of automatic actions for system maintenance

Extending, administrative operations

The last group of administrative operations serve to extend system functions using productive operations. That includes the following:

- extension installations
- configuration of the archive system
- setup and execution of automatic actions for data imports
- setup and execution of automatic actions for document editing
- implementing HL7 servers
- implementing Dicom servers

Preparing the Productive Application

Installation

Installation processes are described in detail in the chapter 'Installation' of this handbook. The handbook 'Installation Description' offers provides you with a stepwise process description of single installation steps. The following briefly summarizes the system installation.

After logistically prepare the computer environment, an installation is the first step for putting an enaio® system into operation. There are various installation processes as the target system to be created may have very different forms. The flexibility of enaio® offers you to either configure a simple system for a single workstation or a complex one in an entire network.

Systems for single workstations require at first the installation of an application server and then the local client installation. After that, further administrative operations follow.

Simple systems consists of an application server, a file server, and one or more clients. As a first step, you need to install an application server. You can run the installation of the file server and the client after server start. At the same time other administrative operations, such as creating object definitions, are performed, further clients can be installed.

More complex systems have more than one application server in multiple server groups. The number of clients is considerably higher. Here, the installation process is much more complex, nevertheless, as a first step you must install an application server. The other application servers can be installed afterwards or at a later time. The file server assigned to a server group cannot be installed until at least one server of the group has been installed and started. After having set up the administrative client, you can continue with further administrative operations. Afterwards or at the same time, you can run the network installation of the clients.

During installation, a wizard will guide you offering dialog boxes. According to the decisions made during the installation process, different installation steps are shown. The administrator is provided with two important tools facilitating the installation. The first is the maintenance mode accelerating repair and update processes, the second is the silent setup enabling automatic installation of the client.

Creating the Object Definition

The definition of DMS objects is an indispensable condition for the operation of an enaio® system. Without defined objects, no sensible system function would be available. Full system performance is not available until having created an adequate object definition.

Start enaio® editor to create an object definition. To do so, enaio® editor must be installed, the application server must run and the workstation at which enaio® editor is running must be licensed accordingly. The logged in user must be provided with the adequate authorizations that allow him to start enaio® editor, edit the object definition and synchronize the database in order that he can execute all actions described.

After initial start, the graphical editor does not offer any object definition. There are two options available:

- You create a new object definition.
- You import an available object definition file.

In the second case, an object definition of another enaio® system is available or the consulting department of OPTIMAL SYSTEMS has prepared one. There will not be further descriptions for this case as some processing steps are left out. The creation of a new object definition is described completely.

Follow these steps to create a new object definition:

- Open the current object definition. Its structure tree showing all objects will be empty as the definition is empty, as well.
- Create a new cabinet.
- Configure the index fields for the cabinet.
- Set up the field properties of the cabinet.
- If necessary, create a new register in the cabinet.
- Configure the index fields for the register.
- Set up the field properties of the register.
- Repeat the register creation process as often as required.
- Create a new document type.
- Configure the index fields for the document type.
- Set up the field properties of the document type.
- Repeat the document type creation process as often as required.
- Repeat the creation process of cabinets with registers and documents as often as required.
- Synchronize the database.

Afterwards, the objects will be available. In the database all necessary root, register, and object tables are created, whereas the object definition file contains information on data sheets, fields, and field properties. To use the objects, start the application server or at least restart the respective executors. (This can be done automatically by the enaio® editor). Furthermore, you need to configure some more settings in enaio® administrator.

Object types

Any type of objects can be created for the object definition. There are different object types available: cabinet, register and document. The following table lists all currently available object types.

Object type	Object type name
0	Cabinet
99	Register
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)

Table 37: Object types

In addition, there are module-spanning document types for which you do not specify the object type until the document has been created definitely. Module-spanning document types can be created as object types from 1 to 5.

Dialog forms

For each object type to be designed, a dialog form is defined. This form offers three different fields of application in enaio® client:

- When creating new objects.
- When searching for objects. In this case, the dialog is called 'query'.
- When displaying object index data. In this case, the dialog is called 'data sheet'.

You can set up different properties for dialog forms. This includes position, size, flags for full text indexing and many more as well as further settings, e.g. background image and object icon.

Fields

Dialog forms offer different types of dialog elements. Next to visual elements, such as a background image and other images, these especially are fields that allow the user to enter data.

The following dialog elements can be integrated into a dialog form:

- Text field
- Check box
- Radio buttons
- Group box
- Button
- Static text
- Image
- Table
- Page control

You can choose any position and size for these fields. It is also possible to assign numerous properties to fields that enhance the object definition's flexibility.

The enaio® editor handbook offers the properties and its meanings in detail.

Further dialog elements

The field properties allow you to display further dialog elements: the catalogs. The following catalogs are available:

- List
- Tree
- Hierarchy
- Database
- Structure
- AddOns:
  - Query AddOn
  - AddOn for database index entry
  - Date AddOn
  - Web AddOn
  - VB Script AddOn
  - Euro AddOn
  - User AddOn
  - User group AddOn

Catalogs are strongly assigned to fields and offer functions regarding the amount of values and field values. Catalogs have the following functions:

- Entry of values into fields:
- Query AddOn
- AddOn for database index entry
- Date AddOn
- VB Script AddOn
- User AddOn
- User group AddOn
- Changing of data, e.g. conversion of currencies:
- Euro AddOn
- VB Script AddOn
- Filtering of the data to be displayed in a field:
- List
- Tree
- Hierarchy
- Database
- Structure
- VB Script AddOn
- Further functions:
- Program control (VB Script AddOn)
- Opening Web pages, sending e-mails (Web AddOn)

Due to its high flexibility, the VB Script AddOn can be applied for all catalog functions. As you can use VB script to access the entire object model of enaio®, further functions may be simply conceivable.

More information on the creation of object definitions can be found in the enaio® editor handbook.

Configuration of the Security System

After having created an object definition and synchronized the database, the next step includes the configuration of the security system in enaio® administrator. Actually, the order can be reverse: first, you configure the security system and then create the object definition. This order is much more inconvenient as permission rights for enaio® objects cannot be set until the object's creation requiring the re-editing of the security system.

The security system must be configured to allow you to provide users with permission rights enabling them to edit created objects and to use installed enaio® applications. The creation of a profile user supports you when setting up a great number of users. The profile user settings can be simply assigned to new users.

The following configuration order of the security system is recommended:

1. creation of a user group
2. assignment of object rights to a user group
3. creation of logical expressions for object rights
4. creation of necessary groups within the steps 1 to 3
5. creation of a user
6. providing the user with system roles
7. assignment of the user to one or more system groups
8. creation of necessary users within the steps 5 to 7

This configuration order is a recommendation. It is also possible to create a user group first and then all users to be assigned to this group. Afterwards, you can create further groups according to the same procedure. In doing so, all users that are meant to be a member in more than one group must be edited multiple times or only once if all necessary groups have already been created. You can extend or change the security system at any time.

When installing enaio®, the user 'root' and the user group 'standard' is created automatically. The user 'root' has administrative rights and is a member of the group 'standard'. It is recommended to create a new administrator and to remove the user 'root'. At least, you should instantly change the password of the user 'root'. Please keep in mind that you cannot remove the last user having administrative rights.

In case you cannot start any administrative application after having installed enaio® due to the fact that you do not know the password of the user 'root', please contact OPTIMAL SYSTEMS support department.

User groups and object rights

The authorization which allows users to edit DMS objects is assigned to user groups. User groups therefore include a number of users with identical rights to a number of DMS objects. Each user can be a member of more than one user group. In case a user is a member of multiple groups which have rights cancelling each other, wide-ranging rights are always valid and complementary rights are added. Users with multiple group membership do always receive the rights of both groups. This applies to logical expressions which can be created for object rights, as well.

Group memberships are defined at user creation and can be changed afterwards. Before users can start being productive, e.g. to edit DMS objects, you must create user groups and distribute object rights. Administrative operations do not require the assignment of object rights.

It would be advantageous to recreate the structure of the organizational unit enaio® is applied in. If, for example, enaio® is applied in a hospital, the creation of ward-related user groups is recommended. At the same time, it may be necessary to create one or more user groups with more extensive rights for users of different stations, such as senior and chief physicians. However, these are just recommendations for the user group creation. For more information on user group design, please contact the OPTIMAL SYSTEMS consulting department.

The security system offers the possibility to synchronize the enaio® user groups with user groups of NT domains. User group names can thereby be imported from the NT domain the administrator is logged on to. This option can be found in the user group setup of enaio®.

Only the group names can be synchronized as enaio® authorizations cannot be derived from domain authorizations. After synchronization, you must configure these groups.

Object rights

The coverage of rights depends on the object type folder, register, and document. The following object rights can be set for folders in enaio®:

- Show index data (R). The indexing of the folder or register can be displayed if this right was granted. This right is a requirement for all other rights. If a user does not have the right 'Show index data' to access folders, the respective search form will not be shown in the object search area. Search forms which are used to retrieve registers and document types contained in folders will not be displayed either. Thus, users cannot retrieve any content of none of the folders.
- Write index data (W). The indexing of the created folders can be created or changed if this right was granted. This right is a requirement for the rights D, X, and U. The user can access registers and documents inside these folders given that he was provided with the respective rights for registers and document types and he is capable of either accessing the folder content over follow-up or workflow, saved queries, portfolios, and e-mail.
- Delete object (D). Folders can be deleted if this right was granted. Users with the 'delete object' folder access right can only delete a folder if they also have the right to delete all objects within the register.
- Output object (X). The folder can be opened, the contents can be displayed if this right was granted.
- Write object (U). With no function for folders and registers.

The following object rights can be set for registers in enaio®:

- Show index data (R). The indexing of the folder or register can be displayed if this right was granted. This right is a requirement for all other rights. If a user does not have the right 'Show index data' to access registers, the respective search form will not be shown in the object search area. The contents of register# can be queried, though, if the user has the respective rights for the registers or documents inside the register.

- Write index data (W). The indexing of the created registers can be created or changed if this right was granted. This right is a requirement for the rights D, X, and U.
- Delete object (D). Registers can be deleted if this right was granted. Users with the 'delete object' file access right can only delete a register if they also have the right to delete all objects within the register.
- Output object (X). The register can be opened, the contents can be displayed if this right was granted.
- Write object (U). With no function for folders and registers.

The following object rights can be set for document types in enaio®:

- Show index data (R). The indexing of the created documents can be displayed if this right was granted. This right is a requirement for all other rights. If a user does not have the right 'Show index data' to access documents, the respective search form will not be shown in the object search area. Access to these documents is neither possible via follow-ups, nor e-mails, saved queries, portfolios or workflow.
- Write index data (W). The indexing of the created document can be created or changed if this right was granted. This right is a requirement for the rights D, X and U. To create a document with pages, the 'write object' access right is also required.
- Delete object (D). Documents can be deleted if this right was granted.
- Output object (X). Documents can be opened, printed and exported if this right was granted.
- Write object (U). The document can be created, document files can be created and overwritten.
- Group annotations (G). Annotations can be created for document types which are only visible for the user group of the user. This right is only relevant for image documents, i.e. D, P and X document types.
- Public annotations (P). Annotations can be created for document types which are only visible for all users who are allowed to display the document. This right is only relevant for image documents, i.e. D, P and X document types.

In case a user has the right 'write object' (U), but neither the right 'group annotations' (G) nor 'public annotations' (P), there will be a problem with static layers. Static layers must not be edited by the user. When the user opens a document, layers will be fixed firmly into the document file. If now the user files this document, the original file is changed. You can handle this problem best by granting the annotations rights to all users that are allowed to edit the documents.

Logical expressions for object rights

Each right described in the 'object rights' section can be limited with logical expressions. The object right can be set up for all objects of the selected object type. Logical expressions are designed in such a way that together with the system variables of the user session and the indexing of a specific DMS object they form a Boolean value. In case this value is true, the object right becomes effective, in case the value is false, the right for this object is denied. The following simply exemplifies this content.

Example: user B is a member of the group G and the right 'show index data (R)' was only granted for the object type O. This right is limited through a logical expression. The logical expression is very simple:

```
creator = #BENUTZER#
```

'Creator' is a field on the object's data sheet, #user# is a system variable with the name of the logged in user. The logical expression signifies that users of the group G can only view the indexing of objects of the type O if the field 'user' contains the name of the logged in user.

Simple logical expressions follow this syntax:

```
field name operator value
```

For field names you can choose all fields of the object's data sheet. Apart from these field names, you can use the following values:

Value	Meaning
#BESITZER#	Owner of the current object.
#COMPUTER-IP#	IP address of the computer the current user is logged on to.
#COMPUTER-GUID#	GUID of the computer the current user is logged on to.
#COMPUTER-NAME#	Name of the computer the current user is logged on to.

Table 38: Fields in logical expressions

For document types, you can also use basic parameter data:

Value	Meaning
#ANLAGEDATUM#	Date of the object creation.
#ANLEGER#	Creator of the object.
#ARCHIVAR#	The most recent object editor. Not changeable after archiving.
#ARCHIVERUNGSDATUM#	The most recent editing date. Not changeable after archiving.

Table 39: additional fields in logical expressions for documents

The following operators are available:

Operator	Meaning
=	Equals
<>	Not equals
<	Less than
>	Greater than
<=	Less than or equals
>=	Greater than or equals
In	Contained in
!in	Not contained in

Table 40: References between fields and values

You can manually enter a constant value or one of the following variables:

Value	Meaning
#BENUTZER#	Name of the current user.
#DATUM#	The current date.
#GRUPPEN#	The groups the user is a member of. Use the operator 'in' or '!in' here.
#COMPUTER-IP#	IP address of the computer the current user is logged on to.

#COMPUTER-GUID#	GUID of the computer the current user is logged on to.
#COMPUTER-NAME#	Name of the computer the current user is logged on to.
#NULL#	The respective index data field is empty.

Table 41: Values in logical expressions

For document types, you can also use basic parameter data:

Value	Meaning
#ANLAGEDATUM#	Date of the object creation.
#ANLEGER#	Creator of the object.
#ARCHIVAR#	The most recent object editor. Not changeable after archiving.
#ARCHIVERUNGSDATUM#	The most recent editing date. Not changeable after archiving.

Table 42: Additional values in logical expressions for documents

For manual value entry you can use wildcards: '*' for any type and number of characters and '?' for any character.

To combine simple logical expressions, use logical operators.

Value	Meaning
AND	logical AND
OR	logical OR
NOT	negation

Table 43: logical operators in complex expressions

The NOT operator stands before a logical expression and is performed first. The operators AND and OR are sequentially performed from left to right. Group the logical expressions with brackets to change the precedence of operators. Therewith, you can create complex and combined expressions, such as the following:

expression1 `AND` (expression2 `OR NOT` expression3)

User and system roles

The right to execute enaio® applications and to perform specific operations within these applications are not granted to user groups but to single users. In enaio® these rights are called system roles. Different system roles apply to different client applications. The appendix offers an overview over all system roles available in the system.

LDAP

The security system offers the possibility to authenticate users through an external LDAP directory service. LDAP stands for 'lightweight directory access protocol'. Given that this type of authentication is enabled, user passwords will not be saved in the DMS database anymore but administered by the LDAP service only.

Users registered in the LDAP service can be imported into the enaio® security system. To do so, at first set up the connection between the entire enaio® system and the LDAP service. You have to set up the following:

- LDAP server: enter the computer name the LDAP directory service is running on.
- Port: this is the IP port enabling communication with the LDAP server.

- Binding string: specific string describing a particular position within the LDAP directory service structure and forming the starting point for queries to the service.
- Mapping: often cryptic looking variable names of the LDAP service are converted to enaio® names that are easier to recognize. These names are used for queries to the directory service.

After configuration, the LDAP service can be used for user import into the enaio® user administration. In the LDAP user administration of enaio® you can execute the following:

- Queries to the LDAP service. Filtered searches consequently are available to LDAP users.
- Import of LDAP users. Found users can be imported into the security system.
- Synchronization with the LDAP service. The enaio® security system does not recognizes automatically whether users have been removed from the LDAP service. That is why you must run the synchronization with the LDAP service.

As described, you must configure LDAP users due to enaio® rights that cannot be derived from the LDAP rights. This includes the assignment to user group and the grant of system roles.
If LDAP authorization has been selected, and the LDAP directory service is not available, no user will be able to log in. Therefore, it is recommended that you create at least one user that has the system role 'Supervisor' and an OS password. In case the LDAP directory service is not available, this user can start enaio® administrator or enaio® enterprise-manager and change the settings of the user administration. It is still necessary to disable LDAP authorization first. This is done by changing the value of the string `LoginMode` (found under the Windows registry key `...\Schemata\4.0\Login`) from '1' to '0'. Then, all users with OS passwords will be able to start the programs according to their system roles.

Profiles

Create a profile user to simplify the setup of a great number of users. To do so, create a user according to the known procedure except that he receives the property 'Profile user'. Assign this user to user groups and provide him with system roles. Afterwards, log in this user to the respective applications and configure all settings to be distributed to other users later.

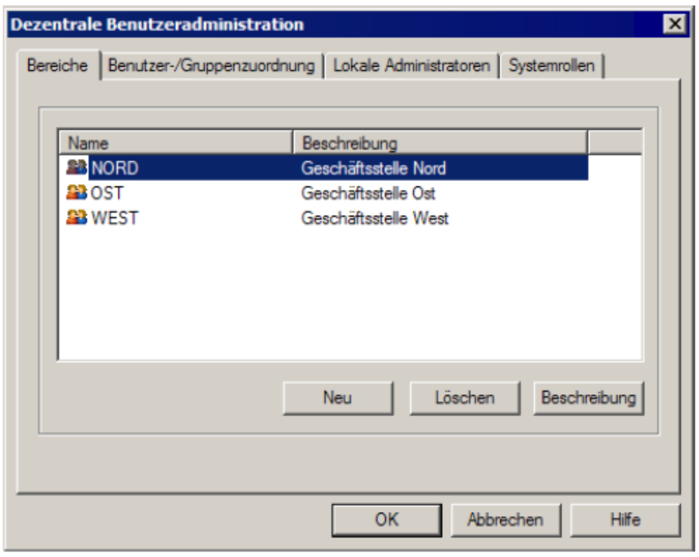
You can distribute the profile after having configured all necessary settings. To do so, assign this user to new or existing users. This task is done by the administrator, .i.e. he must explicitly assign the profile to each single target user.

The following settings of the profile user can be assigned to new users:

- System roles
- Group membership
- Settings made in enaio® client,
- Saved queries
- Extended queries
- Links to external programs,
- Settings for the workspace
- Settings for the navigation
- Scan settings

Remote User Administration

Within complex environments, it might make sense to set up areas for remote user administration through a local administrator.



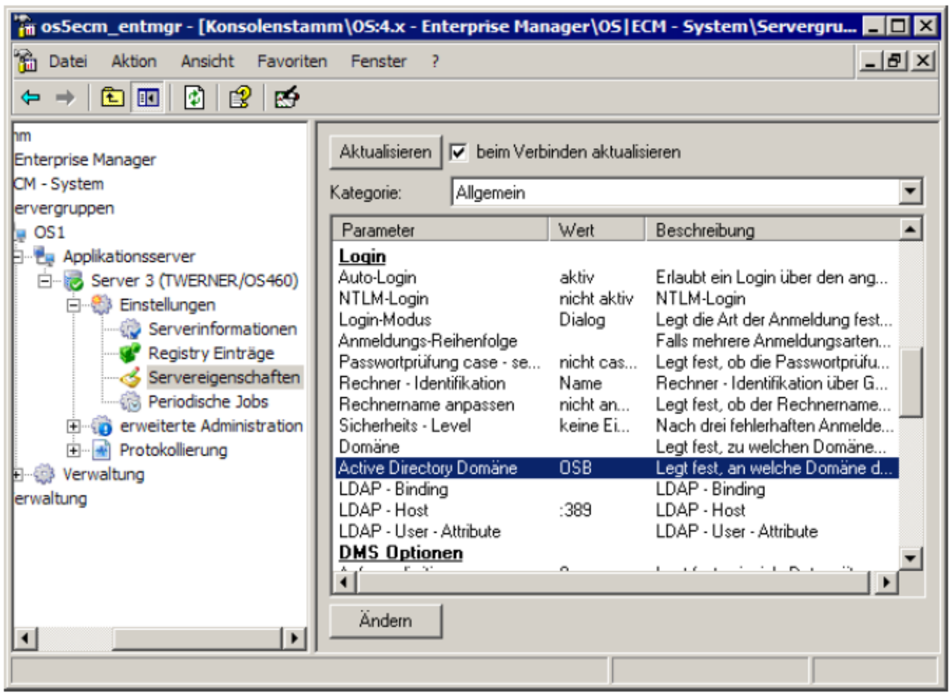
Name	Beschreibung
NORD	Geschäftsstelle Nord
OST	Geschäftsstelle Ost
WEST	Geschäftsstelle West

Figure 55: Remote user administration

In enaio® administrator the system administrator sets up local areas and grants access rights to the group of this areas, defines the person(s) to be allowed to administer local areas, and decides which system roles can be granted by local administrators to the users of their areas. A local administrator can admit users to his local area, assign groups with predefined access rights to users, and grant predefined system roles. The local administrator has the same functions and tasks like the system administrator except that available features and options are limited.

Active Directory Service

Authentication over an Active Directory service is also available. In enaio® enterprise-manager you must define 'Active Directory' as the login mode. Additionally, you have to specify the domain to which enaio® server logs the user on.


os5ecm_entmgr - [Konsolenstamm\OS:4.x - Enterprise Manager\OS|ECM - System\Servergru...

Datei Aktion Ansicht Favoriten Fenster ?

- m
- Enterprise Manager
- CM - System
- ervergruppen
- OS1
  - Applikationsserver
    - Server 3 (TWERNER/OS460)
      - Einstellungen
        - Serverinformationen
        - Registry Einträge
        - Servereigenschaften
        - Periodische Jobs
      - erweiterte Administration
      - Protokollierung
  - Verwaltung
- erwaltung

Aktualisieren ☑ beim Verbinden aktualisieren

Kategorie: Allgemein

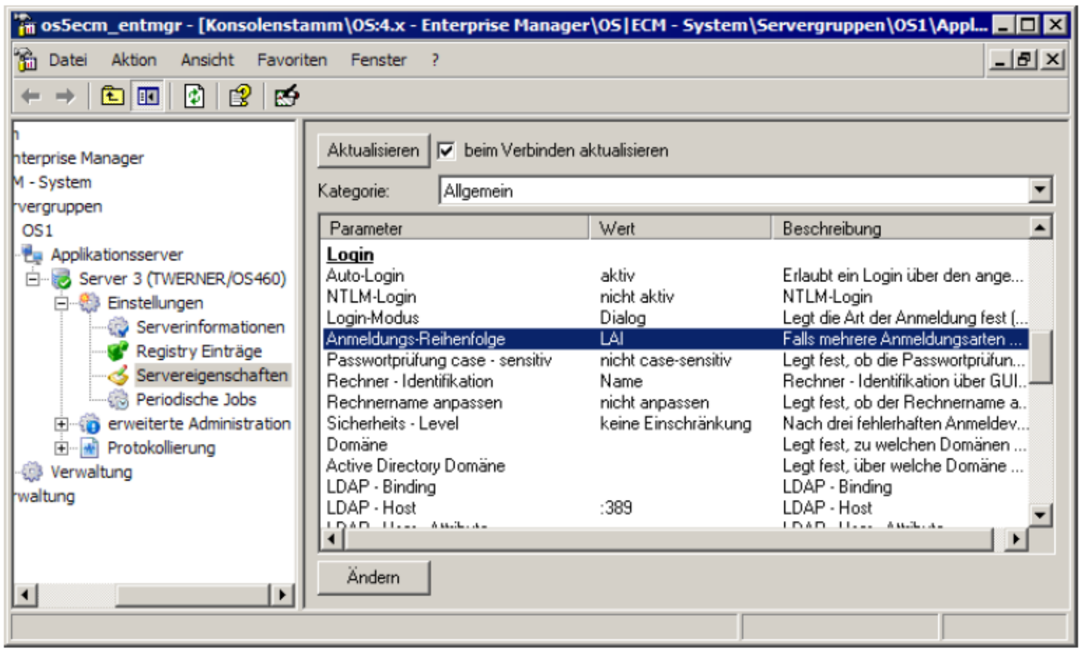
Parameter	Wert	Beschreibung
Login		
Auto-Login	aktiv	Erlaubt ein Login über den ang...
NTLM-Login	nicht aktiv	NTLM-Login
Login-Modus	Dialog	Legt die Art der Anmeldung fest...
Anmeldungs-Reihenfolge		Falls mehrere Anmeldungsarten...
Passwortprüfung case - se...	nicht cas...	Legt fest, ob die Passwortprüfu...
Rechner - Identifikation	Name	Rechner - Identifikation über G...
Rechnername anpassen	nicht an...	Legt fest, ob der Rechnername...
Sicherheits - Level	keine Ei...	Nach drei fehlerhaften Anmelde...
Domäne		Legt fest, zu welchen Domäne...
Active Directory Domäne	OSB	Legt fest, an welche Domäne d...
LDAP - Binding		LDAP - Binding
LDAP - Host	:389	LDAP - Host
LDAP - User - Attribute		LDAP - User - Attribute
DMS Optionen		

Ändern

Figure 56: Active Directory service

Authentication Order

In complex environments it may be useful to apply more than one authentication mechanism. In such a case, you have to define the authentication order. If one authentication to a system fails, further authentications are executed according to the specified order. You can define the authentication order in enaio® enterprise-manager.


os5ecm_entmgr - [Konsolenstamm\OS:4.x - Enterprise Manager\OS|ECM - System\Servergruppen\OS1\Appl...

Datei Aktion Ansicht Favoriten Fenster ?

- n
- nterprise Manager
- M - System
- vergruppen
- OS1
  - Applikationsserver
    - Server 3 (TWERNER/OS460)
      - Einstellungen
        - Serverinformationen
        - Registry Einträge
        - Servereigenschaften
        - Periodische Jobs
      - erweiterte Administration
      - Protokollierung
  - Verwaltung
- waltung

Aktualisieren ☑ beim Verbinden aktualisieren

Kategorie: Allgemein

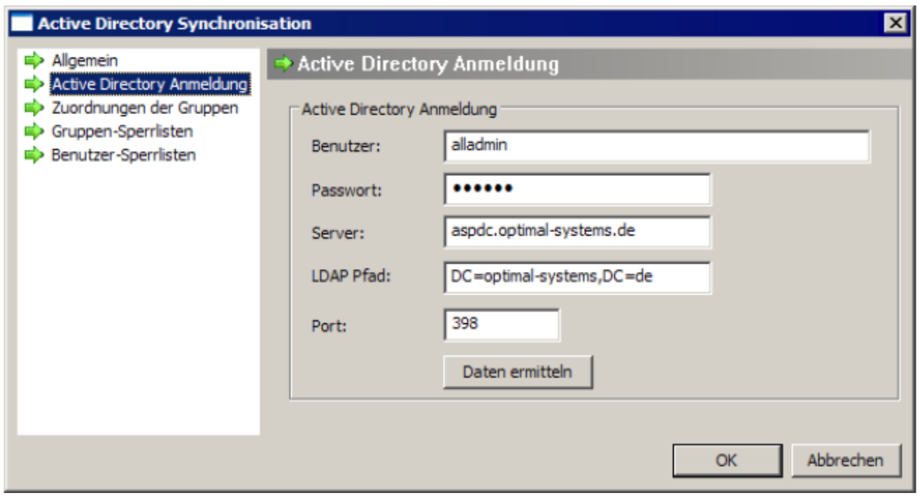
Parameter	Wert	Beschreibung
Login		
Auto-Login	aktiv	Erlaubt ein Login über den ange...
NTLM-Login	nicht aktiv	NTLM-Login
Login-Modus	Dialog	Legt die Art der Anmeldung fest (...
Anmeldungs-Reihenfolge	LAI	Falls mehrere Anmeldungsarten ...
Passwortprüfung case - sensitiv	nicht case-sensitiv	Legt fest, ob die Passwortprüfun...
Rechner - Identifikation	Name	Rechner - Identifikation über GUI...
Rechnername anpassen	nicht anpassen	Legt fest, ob der Rechnername a...
Sicherheits - Level	keine Einschränkung	Nach drei fehlerhaften Anmeldev...
Domäne		Legt fest, zu welchen Domänen ...
Active Directory Domäne		Legt fest, über welche Domäne ...
LDAP - Binding		LDAP - Binding
LDAP - Host	:389	LDAP - Host

Ändern

Figure 57: Authentication Order

Active Directory Synchronization

With the automatic action 'Active Directory synchronization', you can synchronize the data of the OS user administration with the data of an Active Directory user administration.


Active Directory Synchronisation

Active Directory Anmeldung

Active Directory Anmeldung

Benutzer:	alladmin
Passwort:	••••••
Server:	aspdc.optimal-systems.de
LDAP Pfad:	DC=optimal-systems,DC=de
Port:	398

Figure 58: Active Directory synchronization

Active directory groups are converted into OS groups, and existing OS groups are mapped to Active Directory groups. Changes within the Active Directory user administration are also applied to the OS user administration through this service.

At the same time, individual groups and users can be excluded by using block lists. To use this action, include the library `'axacuimp.dll'`. This library does also contain functions for the action 'User and group import'.

Configuration of the License System

When configuring the license system, you can define at which workstations which application and modules are applied. Licenses control whether or not enaio® applications can be executed or not.

All licenses are created and distributed by OPTIMAL SYSTEMS. They are purchased in form of the encrypted file `aslic.dat`. With an import tool this file is imported into the DMS database and filed there in a BLOB field. BLOB signifies 'binary large object'. Instead of simple values, such fields contain complex objects, e.g. files.

The license file import into the database table `oslicense` registers all licenses available in the file `aslic.dat`. These licenses, called modules, must be distributed to designated workstations. For this purpose, the license management of enaio® enterprise-manager offers a tool. Therein, configure the following:

- At first, stations are added. Stations can be imported from existing NT domains or created manually.
- Add modules to existing stations.

When finishing license system configuration, all information are written to the database. After that and having saved all license system changes, the corresponding modules are available for the configured stations. This applies regardless of the users as licenses refer to stations.

The TCP/IP address, GUID number, or the computer name is used to identify workstations. The computer name is determined through the domain controller or the file 'hosts', whereas the TCP/IP address or the GUID number are always used internally. Configured, logical enaio® workstations are saved in the table `osstations`. The physical GUID and IP addresses of each station are saved in the table `osstationnc` and assigned to the logical stations. The available network adapters of the workstation prescribe these addresses. So it is possible to manage stations with more than one network adapter. The table `oslicresources` connects both tables `osresources` and `osstations`. Here, available resources are assigned to the single stations.

During operation, the application server analyzes available and requested licenses. If the maximum is reached, licenses will not be granted anymore and the component cannot be launched. In the table `ossession` all sessions, i.e. connections of client applications to enaio® servers, are saved. Each client login to the server is saved as a unique entry into this table. So, the session is distinctively connected with a server and a station. Resources being requested and occupied by a session are saved into the table `oslockedres`. This table is connected with both tables `ossession` and `osresources`.

Only client connections within a valid session can occupy resources, such as licenses. In case enaio® server has been terminated irregularly both the session entries and the occupied resource entries remain entered in order to enable another server of the same server group to take over this session.

Distribution possibilities for licenses on workstations depend on the license type. A module can be licensed in two different ways:

- Named: a module with such a license can only be executed on exactly one station. 10 named licenses can be executed in exactly 10 stations.
- Concurrent: concurrent licenses enable the module execution on any number or computers as long as the maximum number of licenses has not been exceeded. 10 concurrent licenses can be executed on about 100 stations but only on 10 stations at a time.

Some modules require the entry of configurations in licenses, e.g. for automatic import actions. The entered number of configurations cannot be exceeded. This entry is used for other modules to control automatic license release. The entry is read as a number of minutes. A license is automatically released when an application was idle during the specified time.

License import

The first license import is performed when installing the first application server. Without any valid license the application server cannot start. Test licenses for temporary use can be used. During installation, the specified license file `aslic.dat` is imported completely into the database, the enclosed file `aslic.cfg` is analyzed, and the content is written to different database tables. Into the file `aslic.cfg` a default module distribution is entered generally providing the assignment of the application server license to the specified server station. The application server can run only on a GUID-named station. Before license creation, you have to communicate this GUID to OPTIMAL SYSTEMS.

Newly purchased licenses are provided by OPTIMAL SYSTEMS in form of the file `aslic.dat`. This file must be imported into the database using enaio® enterprise-manager.

With enaio® enterprise-manager you set up an ODBC connection by specifying the name of the ODBC source, the user name, the password of an authorized database user, and the directory the license file is located in. Then, you can import the license file.

Setting up Document Types

For some available object types you need to configure further settings after having created objects in enaio® editor, e.g. the dragging & dropping behavior of e-mails or viewer setups.

Some of the introduced object types offer integrated viewers in enaio® client. For explanation, the table with available object types is again displayed here.

Object type	Object type name
0	Cabinet
99	Register
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)

Table 44: Object types

The object types 'cabinet' and 'register' do not need a viewer as only their indexing and their content is displayed. This is done from enaio® client. Further administrative settings are not required. This applies to all document types defined as pure references only, i.e. they are saved without any document file. Viewers implemented in enaio® client display X-, D-, and P-document types. That is why the viewers do not need to be configured.

W-Documents are not provided with integrated viewers as they are characterized through the required connection to Windows applications. Therefore, you must configure viewer and templates for W-Documents. Furthermore ensure that applications necessary for viewing and editing of W-Documents are installed at respective stations.

enaio® client uses functions of the Windows Media Player to display W-Documents without requiring additional viewer configuration. Nevertheless, the Windows Media Player must be correctly installed.

E-Documents can be displayed internally or over MS Outlook. MS Outlook Express is not supported. The format of the selected e-mail defines the display type. If you insert e-mails from the internal inbox of enaio® client as DMS objects, they are filed in MIME format. In this case, these e-mails are displayed over the internal viewer. If you drag and drop e-mails to import them from MS Outlook, they are displayed over MS Outlook. In both cases a viewer configuration is required. In the last case, you just need to ensure that MS Outlook has been installed in the workstations such e-mails should be displayed on. You must manually configure the drag & drop behavior of e-mails:

XML Documents are displayed through the display engine of the Internet Explorer. The installation of the Internet Explorer version 6 is recommended. In addition, XML documents are not displayed correctly until style sheet templates have been prepared before. You furthermore need to register these templates by manually entering them into the file `as.cfg`.

More information on the configuration of document types can be found in the enaio® administrator handbook.

Events and Scripting (Event Editor, AddOn and Index Scripts)

All core components of enaio® are written in Visual C++ ensuring high performance and flexibility. Some additional components are written in Visual

Basic (VB). Visual Basic enables efficient creation of powerful applications with special orientation towards GUI functions.

At multiple points in the enaio® system, script language is applied: Visual Basic Script (VB script). VB script offers almost the same features as VB does. The essential difference is that VB script does not require compiling and, provided that the Windows scripting host is installed, is interpreted in the runtime environment.

With VB script you can integrate minor program functions for different occasions in enaio® to customize the respective application feature. In addition, you can realize a great number of functions that are not implemented into the core component as it would not be cost-effective.

On the following system parts you can create and run scripts:

- Events in enaio® client
- Events in enaio® server
- VB Script AddOn in enaio® client
- Precheck and aftercheck scripts in enaio® capture
- Workflow Events
- Automatic action data/document import
- Automatic action data/document export

The following summarizes the application of VB script in this environment. The handbooks for enaio® client, enaio® capture, workflow, and enaio® administrator offer more detailed descriptions.

Events

Events are integrated scripts that are run as soon as specific user actions in the client or the server occur. Here, client scripts will be briefly introduced. For information on the implementation of server scripts please read the respective sections of the enaio® server handbook. Client scripts can be run, e.g. when launching enaio® client, opening a data sheet, or when moving an object. When such an event occurs, enaio® client checks whether or not a script for the occurred event has been assigned to the respective object. If so, the respective script is executed.

You can use these events to validate and change data. Also, you can start external programs. It is possible to use the complete VB script command set in scripts.

In the event editor window of enaio® client events are assigned to available DMS objects and edited. An EVE license is required to run the event editor in enaio® client. The following enaio® client entry points are realized for these events:

Event	Reference	Time of the execution
AfterFinishQuery	Query form	After executing of the query
BeforeStartQuery	Query form	After the user pressed 'Start query' on the query form, before the query is actually executed.
AfterLink	Client	After a notes link was created.
BeforeLink	Client	Before a notes link was created.
AfterDeleteLink	Client	After a notes link was deleted.
BeforeDeleteLink	Client	Before a notes link was deleted.
AfterLogin	Client	After the user has logged on.
BeforeLogout	Client	Before the user has logged out.
OnCloseApp	Client	After check in of all documents before exiting the client.

OnStartApp	Client	After the client start, right before 'AfterLogin'.
OnTime	Client	When a specific time was reached.
AfterSave	Data sheets	After the plausibility check by the client and saving.
BeforeSave	Data sheets	After 'Save' was pressed, before the plausibility check by the client and saving.
OnClickItem	Data sheets	If a linked button is pressed.
OnChangeActivePage	Data sheets	If the active page of the dialog element 'Page control' is switched.
OnShow	Data sheets	After the data sheet was opened and filled with data.
AfterDelete	Hit list	After an object was deleted.
BeforeDelete	Hit list	Before an object is deleted.
BeforeOpen	Hit list	Before a document is opened.
AfterSaveDocument	Hit list	After a document was saved.
BeforeSaveDocument	Hit list	Before a document is saved.
OnCopy	Hit list	If a user drags a register or a document with the mouse to a new location and selects the option 'Create reference copy'.
OnMove	Hit list	When a document or register is moved.

Table 45: Event classes

Event scripts are stored encrypted in the database. Event classes which represent the possible event types are saved in the table `oseventcode`. The table `osevents` contains all events. They can be imported and exported as files.

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

The client requires the return value for the event. The code must contain this value. The value `resultcode=1` saves the data out of the transferred file into the data form, `resultcode=0` leaves the data as they are, `resultcode=-2` imports the data from the transferred file to the data form which remains open with the changed data in the client. The value `WriteToFile` saves all changes in the transferred file.

Export and import of event scripts

OPTIMAL SYSTEMS delivers customer-specifically designed events as encrypted files that after import can be assigned to an archive objects and saved into the database.

It is also possible to import and save single events as encrypted files in enaio® client. You cannot import unencrypted files with the extension `.eve` but view them using a text editor. Encrypted event files have the file extension .`evc` so that opened in a text editor they look like absurd strings. enaio® client additionally allows you to export/import all scripts into the system.

Example for an event script

In this script, the name and field content of an AddOn field is output and then the AddOn field content changed:

```
MsgBox (AddonFields.Item(0).Name + ' = ' + AddonFields.Item(0).Value)
AddonFields.Item(0).Value = 'neuer Wert'
WriteToFile()
```

Further information can be found in the OS_events handbook.

VB script AddOn

VB script is further applied in the client with the VB Script AddOn. This AddOn is a dialog element assigned to a field. In enaio® editor, you can configure it with the field properties. In addition, into the respective section of the file `as.cfg` that the VB Script AddOn is assigned to, the following line has to be entered:

```
EXTRA00=e:\os\vbscripts\scriptname.vbs
```

If the VB Script AddOn is assigned to an index field, the field receives a catalog button. By pressing the left mouse button, the connected script is executed, whereas the right mouse button will start the VB editor.

The VBScript AddOn allows execution of VB scripts, for example, to edit the indexing of fields. If configured in enaio® editor, scripts are automatically run by the VBScript AddOn once a data sheet is closed, and if not, you have to run them manually by clicking the AddOn's catalog button. The same button allows you to edit scripts given that a VB script editor license is available.

VB scripts can be delivered by OPTIMAL SYSTEMS GmbH on request or can be created with the VB editor. The AddOn must be licensed with the module 'VBX' and the VB editor requires the module 'VBE'. For the VB editor, the user needs the right 'enaio® Client: Create events' which is set in enaio® administrator.

Further information can be found in the enaio® editor handbook.

Scripts in enaio® capture

VB script is also applied in the validation component `axvalid.exe` of enaio® capture. In this environment, script should check, filter, and automatically change data. These functions are called precheck and aftercheck, here. Data are tested before opening a new data record and after closing an edited data record. If the test fails, respective data can be changed automatically or data record closure is denied and the user must re-edit the data. Erroneous data can be highlighted in color.

Scripts are configured in the batch options in AXVALID. As above mentioned, used script files are filed encrypted. These scripts are not filed into the DMS database but into the file system.

Further information can be found in the enaio® capture handbook.

Automatic action data/document import

If configured, a minor script executes the automatic action 'data/document import' for each import data set. This script allows you to perform validations being beyond the scope of usual checks within data and document imports.

Without scripts you can only validate whether or not data types are correct. Scripts make content validations, such as plausibility of date specifications, and values, such as replaced expressions through catalog numbers, possible.

Automatic action data/document export

If configured, a minor script executes the automatic action 'data/document export' for each export data set. Scripts make content validations, such as plausibility of date specifications, and values, such as replaced expressions through catalog numbers, possible. In addition, a script can be launched at the end of an export. Actions can therewith be run directly after a successful complete export process.

Annotations to scripting

VB script is a powerful script language being less extensive than Visual Basic. VB script is procedure-orientated and is interpreted, i.e. in contrast to VB, with VB script you cannot write independent, executable files. COM objects can also be created with VB script allowing you to control applications like MS Word, MS Excel and enaio® client. The following command enables COM object creation:

```
MyAS = CreateObject('optimal_AS.application')
```

For reasons of compatibility, the COM object of the clients is called optimal_AS. If there already is an object, i.e. if exactly one enaio® client is running, its COM interface is used. If no client object is available, the most recently registered enaio® client instance starts using its COM interface. If more than one client is running, the object and COM interface of the most recently started client are used.

The object created this was offers various methods to perform the following functions within the client:

- object-related functions
  - searching for objects
  - inserting objects
  - updating objects
  - gathering information on objects
  - opening objects in the client
  - printing objects
  - etc.
- client control
  - activating the client
  - logging in a new user
  - closing all windows
  - requesting and releasing licenses
  - sending e-mails
  - etc.
- more functions
  - gathering information on the client environment
  - converting XML files with style sheets into a target format
  - converting image files into another target format
  - troubleshooting within the COM interface
  - etc.

It is not possible to apply all COM functions in VB script given that in VB script the variable dimensions are limited to the type VARIANT. But COM functions do often require data of the type LONG.

The realization of important API functions is quite difficult if not impossible. That includes, for example, functions to write INI files.

Many commands require inconvenient and laborious realization. That is why some libraries created by OPTIMAL SYSTEMS are installed and registered together with the enaio® installation. They are realized in the following library files:

- `oxactive.dll` offers numerous functions for event execution.
- `oxvbbas.dll` is a library making available extensive help functions for programming in VB and VB script.
- `oxvbscript.dll` provides many COM functions not realizable in VB script.

OXACTIVE.DLL

The integrated ActiveX control `oxactive.dll` offers methods and objects that the data from the handoff file can be read and edited with. The ActiveX control `oxactive.dll` offers the following objects:

- ASFile supports the dealings with both files the AddOn and transferred file for events.
- RequestPages offers a collection of data forms.
- RequestPage represents a data form.
- ASFields offers all fields of a data sheet.
- ASField represents a field of a data sheet.

In addition to that, you can execute actions with VB script independently of enaio® client or communicate with enaio® client over the COM interface independently of the transferred file.

OXVBBAS.DLL

This library supports the programming in VB and VB script. The oxvbbas object offers the following interfaces:

- General functions are simply named 'Functions'. Here, you are informed about the Windows operating system, file versions, file types etc.
- AS_Functions are functions of the enaio® client interface. They support the determination of object, main, and sub types of enaio® objects.
- INI_Functions support the editing of INI files. They allow you to determine single or all keys of a section, read and write values.
- Reg_Functions support the editing of the Windows registry. They allow you to determine, write and delete keys and their values.

OXVBSCRIPT.DLL

The library `oxvbscript.dll` delivered with enaio® provides you with some equivalent general enaio® COM functions not offered by VB script.

- InsertFileList
- UpdateArchiveData
- InsertIntoDocument
- InsertIntoRegister
- InsertIntoArchive
- GetFilesFromIDEx

You can integrate these functions using the call `CreateObject('oxvbscript.Function')`. Parameters and return values do exactly correspond to those of the COM functions. Details on the COM interface can be found in the COM interface handbook.

System Maintenance

Monitoring the Application Server

Besides server logs, a server console can be opened to monitor the operation of the application server. This server console is available if the application server service was started under a local login or if the application server was not started as a service, but as a regular application. The server console offers a number of information and configuration possibilities.

During normal operation the use of the server console is not required. It is useful to determine possible causes of irregularities and should only be used by support personnel or the development department of OPTIMAL SYSTEMS GmbH.

Communication between Server and Client

Server and client communicate with an RPC ('Remote Prodecure Call') protocol. By default a compact and high-performance BIN RPC mechanism is set up. Switch to XML RPC is not recommended. To increase the data throughput the data can be compressed within the BIN or XML RPC mechanism. By default compression is disabled to save computer resources. Mainly for the use in subsidiaries with slow network connections (ISDN) compression should be enabled.

To recognize network problems it may be useful to enable logging at this point. Then the entire network traffic between client and server can be written to a log file for later analysis. The log file 'net_cln.log' is located in the temp directory of the client. This setting will cause high performance loss on the client side.

To configure the communication the file OXMLJSC.CFG must be edited in the installation directory of the client.

The following parameters can be configured (default values are underlined):

lognetwork= 0/1		logging at data transfer level (off/on)
binrpc=0/1	–	switch between BIN RPC/XML RPC
hashed=0/1		generate and send a hash value of the job call data but not of the files. The function is enabled and, to disable it, you must set the value to '0'.
packed=0/1	–	Data of the job call are compressed if a size limit is surpassed. The entire RPC packet is compressed for the server. This compression is irrelevant for the file transfer. The function is disabled and, to enable it, you must set the value to '1'.
packbound=300		size limit for job size in Byte
rowbase64=0	–	only BIN RPC: Mime-encoded parameter in job calls (parameter type 6) are transferred directly, i.e. not Mime-encoded
mac=0/1/2/3	–	Determine IP address for a MAC address. 0 – Determine with Netbios, if not successful, determine with Ip-Helper 1 – Determine with Netbios only 2 – Determine with Ip-Helper, if not successful, determine with Netbios 3 – Determine with Ip-Helper only

During installation the file `oxmljsc.cfg` is copied to the directories `\client32\` and `\admin\`, and it has the following entry:

```
mac=3
```

Automatic Actions

Automatic actions are tools used to perform essential system tasks. This includes import and export of data, system maintenance and document processing functions. Also, there are functions to control automatic actions. A detailed description can be found in the documentation on automatic actions.

The following automatic actions are available:

- Import and export of data and document files
  - ASCOLD-Import, library `axcold.dll`. This action is used for the import of COLD data. COLD stands for Computer Output on Laser Disk. This procedure is common for mainframe computer installations.
  - ASFax Import, library `axfax.dll`. This action is used for the import of incoming fax messages.
  - Data/document import, library `axacimp.dll`. This action replaces the import functions of the action 'import/export' and adds the import of XML, Excel, and Access to it. A wizard simplifies the configuration.
  - DICOM import, library `axacdcm.dll`. This action imports DICOM files that are made available, e.g. by the enaio® DICOM import server.
  - IMAGESTREAM import, library `axacimst.dll`. This action is used for the import of IMAGESTREAM data.
  - Data/document export, library `axacexp.dll`. This action replaces the export functions of the action 'import/export' and adds the export of XML, Excel, and Access to it. A wizard simplifies the configuration.
- System maintenance
  - Clean subscription, library `axacabo.dll`. This function allows you to delete subscription entries according to customer demands.
  - Database update, library `axacdbst.dll`. This action updates the database statistics to optimize database performance.
  - Cleanup of configuration and log files. Library `axaccl.dll`. This action deletes configurations from the configuration history and logs according to customer demands.
  - Cleanup of cache, library `axaccach.dll`. This action deletes document files from the server cache. The server cache contains document files of which the originals are either in the WORK area of another server group or stored on archiving media.
  - History maintenance, library `axachist.dll`. This function allows you to delete document history entries according to customer demands.
  - System check, library `axacsys.dll`. Checks for and corrects specific database conditions.
  - Clean follow-up, library `axacwdvl.dll`. This function allows you to delete follow-up entries according to customer demands.
  - Clean trash can, library `axaccleantrash.dll`. This function cleans up the system trash can according to customer demands.

    - User export/import, library `axacuexp.dll/axacuimp.dll`. This function allows you to import and export system users. As a result, creating test systems/mirrored systems becomes simpler.
- Document processing
    - Archiving, library `axacarc.dll`. This action transfers documents into archive areas and removes them from the WORK area.
    - Dearchiving, library `axacunac.dll`. This action dearchives and transfers document types into the WORK area of the server.
    - Document retrieval, library `axacpref.dll`. This action retrieves documents from other application servers.
    - Media dearchiving, library `axacunme.dll`. This action dearchives and transfers document types from specific media into the WORK area of the server.
    - Object encryption, library `axaccrypt.dll`. This action encrypts document files of DMS objects.
    - Pagination, libraries `axacpage.dll`. This function paginates document pages, i.e. it adds specific information, such as the page number, to a page.
    - Full text indexing, library `axacidx.dll`. Performs full text indexing on pre-existing DMS objects.
    - XML TagExtraction, library `axacxmle.dll`. Generates delimited ASCII files from XML files.
    - XML transformation, library `axacxmlc.dll`. Performs conversion of XML files into other XML formats.
- Controlling automatic actions
    - Action sequence, integrated in enaio® administrator. Can be used to build sequences of automated actions.
    - COM action interface, library `axaccom.dll`. Can encapsulate COM libraries and make them accessible as automatic actions.
    - Start external program integrated into enaio® administrator. Starts external Windows applications.
    - Start external program from command line, library `axacexec.dll`. Starts external windows applications and batch files, and allows return codes to be read back.
    - Synchronization, library `axacsync.dll`. Enables the synchronization of automatic actions.

Hardware and software requirements

Check conditions before preparing automatic actions. Automatic actions are set up in and are executed by enaio® administrator and by enaio® start. Apart from that some actions require particular hardware and software conditions. For example, audit-proof archiving requires an optical drive for write-once media. Further conditions are mentioned together with the respective automatic action.

Licenses

Some actions can be performed with the enaio® administrator license, other require individual licenses. All licenses are created and distributed by OPTIMAL SYSTEMS. The program `aslicimpexp.exe` registers the licenses in enaio® contained in the imported file `aslic.dat`. In the license system of enaio®

administrator you can distribute the licenses to workstations. The license string 'ADM' is required for configuring and starting automatic action with enaio® administrator. In addition, the license string 'AXA' is required for automatic action execution.

The following lists all available automatic action with corresponding license strings:

Automatic action	License string
Subscription maintenance	ADM
Action sequence	ADM
Database update	ADM
Archive	ARC
ASCOLD Import	COL
ASFax Import	FAX
Cleanup of configuration and log files	ADM
Cache maintenance	ADM
COM action interface	ADM
Data/document import	AIE
Data/document export	AIE
Dearchiving	ADM
DICOM Import	ADM
Document retrieval	ADM
Start external program	ADM
Start external program from command line	ADM
History maintenance	ADM
IMAGESTREAM Import	LIS
Media dearchiving	ADM
Object encryption	KRY
Pagination	PAG
Synchronization	ADM
System check	ADM
Full text indexing	ADM
Follow-up maintenance	ADM
XML Tag Extraction	ADM
XML transformation	ADM

Table 46

Detailed information on the license system can be found in the administration handbook.

The installation of enaio® administrator includes all automatic actions. The several library files are copied into the directory enaio® administrator is installed in. You do not need to register the libraries as they are addressed by enaio® administrator as standard API ddls.

Multilingualism

enaio® is also designed to be multilingual. Thus, it can be applied in international companies and organizations. Generally, enaio® supports multilingualism regarding the user interface and the object definition. Users can customize both areas independently of each other. Users can customize both areas independently of each other. enaio® is available in German (default language), English.

Multilingual object definitions

enaio® editor permits you to create index data forms in several languages. Depending on the user-specific setting in enaio® client the logged in user will then have the data forms displayed in the selected language.

Multilingual documentation and online help

Documentations and online help for enaio® are available in German and, to a great extent, in English. OPTIMAL SYSTEMS provides you with these documentations on the basis of MS Word files and in PDF or HTML format.

Multilingualism in the server

According to the language settings in the server, error messages and warnings the server sends to the clients are translates. The Microsoft Message Compiler is used to enable multilingualism in the server. For each language you can create language files in ASCII format that are compiled in the server.

Also, applications such as enaio® enterprise-manager and OS_protocol-viewer affecting the server settings directly are internationalized in the same way. The GUI language of these modules is selected in the configuration file or over menu entries.

Multilingualism in enaio® office-utilities

The enaio® office-utilities offer multilingualism, as well. These modules' language can be changed over a menu item in the GUI. In the same manner, the compilation of language resources in the respective programs makes multilingualism possible. The language files are also available as ASCII files.

Multilingualism in client applications

enaio®'s first language is German. The enaio® client user settings allow users to switch between the languages English and German. When changing the language in enaio® client, the GUI language of all enaio® client applications that can be internationalized of the logged in user is changed, as well. They will start with the language selected in enaio® client.

Startup Behavior of enaio® client

Extensive registrations are performed and various system and user settings are reloaded from the server when starting enaio® client. Startup can be very slow in networks with insufficient bandwidth performance.

You can start enaio® client with the command line parameter '-slog' to determine the start process chronology. In this mode, the program launch is logged in detail. Open the respective log with the menu item:

FILE › LOGGING › VIEW START LOG

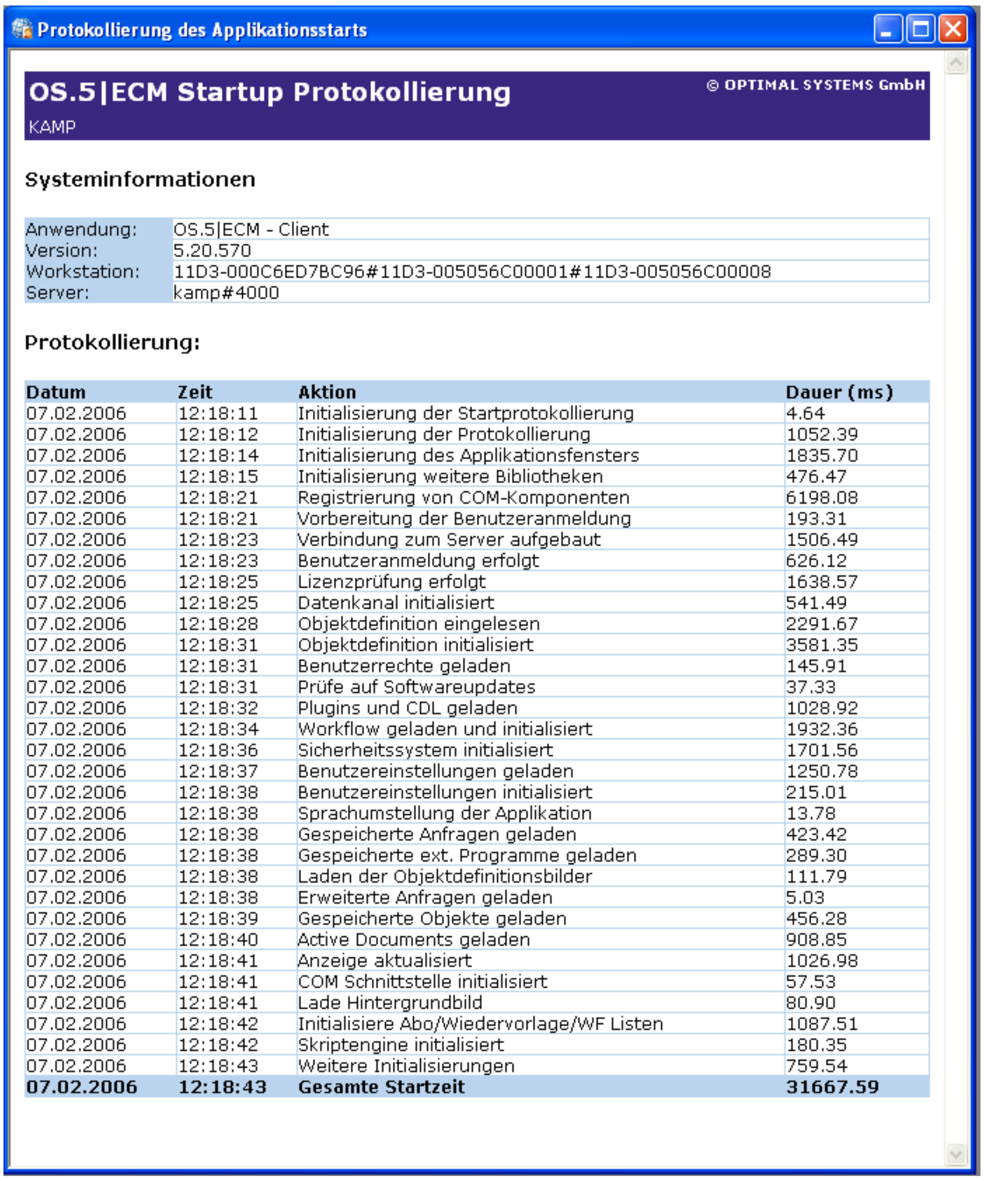
You can also send the log in XML format or as e-mail.

The startup behavior of enaio® client can be controlled in great detail by editing the configuration of the system. Large or extensive object definitions, an extensive rights system with many object expressions can lead to long system startup times.

The following options allow limited optimization:

- Disable program components launched at system start in the registry:
  - The entry NOREGISTER=1 in the section [ARCHIV] of *asinit.cfg* in the installation directory enaio® client.
- Avoid images in the object definition
- Do not use a large wallpaper for the desktop of your enaio® client

- Do not use additional languages in enaio® client
- Optimize follow-ups and subscriptions of the user
- Reduce the number of saved queries and external programs
- Reduce the size of the object definition
- Reduce the size of the rights system.



OS.5|ECM Startup Protokollierung

© OPTIMAL SYSTEMS GmbH

KAMP

Systeminformationen

Anwendung:	OS.5|ECM - Client
Version:	5.20.570
Workstation:	11D3-000C6ED7BC96#11D3-005056C00001#11D3-005056C00008
Server:	kamp#4000

Protokollierung:

Datum	Zeit	Aktion	Dauer (ms)
07.02.2006	12:18:11	Initialisierung der Startprotokollierung	4.64
07.02.2006	12:18:12	Initialisierung der Protokollierung	1052.39
07.02.2006	12:18:14	Initialisierung des Applikationsfensters	1835.70
07.02.2006	12:18:15	Initialisierung weitere Bibliotheken	476.47
07.02.2006	12:18:21	Registrierung von COM-Komponenten	6198.08
07.02.2006	12:18:21	Vorbereitung der Benutzeranmeldung	193.31
07.02.2006	12:18:23	Verbindung zum Server aufgebaut	1506.49
07.02.2006	12:18:23	Benutzeranmeldung erfolgt	626.12
07.02.2006	12:18:25	Lizenzprüfung erfolgt	1638.57
07.02.2006	12:18:25	Datenkanal initialisiert	541.49
07.02.2006	12:18:28	Objektdefinition eingelesen	2291.67
07.02.2006	12:18:31	Objektdefinition initialisiert	3581.35
07.02.2006	12:18:31	Benutzerrechte geladen	145.91
07.02.2006	12:18:31	Prüfe auf Softwareupdates	37.33
07.02.2006	12:18:32	Plugins und CDL geladen	1028.92
07.02.2006	12:18:34	Workflow geladen und initialisiert	1932.36
07.02.2006	12:18:36	Sicherheitssystem initialisiert	1701.56
07.02.2006	12:18:37	Benutzereinstellungen geladen	1250.78
07.02.2006	12:18:38	Benutzereinstellungen initialisiert	215.01
07.02.2006	12:18:38	Sprachumstellung der Applikation	13.78
07.02.2006	12:18:38	Gespeicherte Anfragen geladen	423.42
07.02.2006	12:18:38	Gespeicherte ext. Programme geladen	289.30
07.02.2006	12:18:38	Laden der Objektdefinitionsbilder	111.79
07.02.2006	12:18:38	Erweiterte Anfragen geladen	5.03
07.02.2006	12:18:39	Gespeicherte Objekte geladen	456.28
07.02.2006	12:18:40	Active Documents geladen	908.85
07.02.2006	12:18:41	Anzeige aktualisiert	1026.98
07.02.2006	12:18:41	COM Schnittstelle initialisiert	57.53
07.02.2006	12:18:41	Lade Hintergrundbild	80.90
07.02.2006	12:18:42	Initialisiere Abo/Wiedervorlage/WF Listen	1087.51
07.02.2006	12:18:42	Skriptengine initialisiert	180.35
07.02.2006	12:18:43	Weitere Initialisierungen	759.54
07.02.2006	**12:18:43**	**Gesamte Startzeit**	**31667.59**

Figure 59: Start logging view in enaio® client

Problems during startup of enaio® client can occur if previously started client processes have not been exited properly. In such case users can make sure that these processes are quit with the QUIT OTHER CLIENT PROCESSES AT PROGRAM START setting. This setting can also be adjusted by modifying an entry in the configuration file `etc\as.cfg` for the entire system:
[CLIENT]
KILLPROCESSES=1

Command Line Parameters of enaio® client

Normally, the enaio® client is started without command line parameters. The following command line parameters are available for special requirements:

- -UID: The name of the user who will be automatically signed in to the client can be specified here. If name and password are specified, no login dialog will be displayed when the application is launched.
- -PWD: A password of the above mentioned user can be specified here. If name and password are specified, no login dialog will be displayed when the application is launched.
- -SRV : The server you want to connect to can be specified here. The syntax is identical to the entries in *asinit.cfg*:.
- server#port: Use this parameter to ignore the server entry in asinit.cfg.
- -N: If this command line parameter is set, the splash screen will not be displayed at startup.
- -SLOG: The system start will be logged by the client. This log can be viewed in the client.
- -I: The client will register its program modules. After registering the components the client will be automatically quit.

Example: `ax.exe -uid user -pwd password -srv osserver#4040`

Summary of enaio® Administration Tools

enaio® editor

The following actions are main functions of enaio® editor:

- Creation of DMS objects.
- Editing of input forms to create, modify and search objects. Fields can be created and the tab order of fields can be set.
- Saving the object definition. Information on DMS objects are stored in the object definition and in the database. After editing the objects all changes must be saved.
- Database adaption
- Table check
- Also, prepared entire object definitions or parts of objects definitions can be imported.

Information on how to perform the actions listed above can be found in the enaio® editor handbook.

enaio® administrator

The following actions are main functions of enaio® administrator:

- Setup of the integrated system. Numerous system parameters, e.g. login behavior, adding action DLLs etc. can be set here.
- Configuration of the security system. User and user groups can be created. Individual users can have system roles, i.e. application rights for enaio® applications, user groups are assigned object rights which can be restricted by logical expressions. User profiles can be prepared as templates for new users.
- Setting up document types. Applications and templates can be prepared for W-Documents. These templates are then assigned to applications and users or user groups. Certain parameters must be manually edited for other document types, e.g. e-mail and XML.

- Setting up the archive printer. Background images can be set up for documents which are printed into the DMS.
- Automatic actions. Automatic actions can be set up for different purposes. The following system functions can be performed with the help of automatic actions:
- import and export,
- system maintenance,
- document processing,
- controlling automatic actions.

enaio® enterprise-manager

The following actions are main functions of enaio® enterprise-manager:

- Administration and configuration of enaio® server.
- Configuration of the license system. The purchased licenses can be assigned to the available workstations.
- Configuration of the archiving system. All settings regarding archiving, i.e. writing of document files to optical media, can be set here.
- Viewing of the tables of the system database.

Logging Configuration

Logs can be set up using enaio® manager-for-logfiles (`axprotocolcfg.exe`). It is located in the respective installation directories of server and client components.

The detailed settings capabilities of channel properties allow configuring even logging workstation- and component-specifically.

enaio® editor-for-events

VB scripts can be created in the event editor which is integrated in enaio® client to execute specific actions in enaio® client. The entry points for the scripts are application events, e.g. user login or document related actions, e.g. opening a data sheet.

enaio® capture in Administrative Mode

The following must be performed in administrative mode to prepare enaio® capture for productive use:

- Creation and editing of configurations. Name, batch name and access permissions are specified for the configuration.
- Creation of subprogram types. Individual enaio® subprogram files are prepared to be integrated in the configuration.
- Adding subprograms to the configuration.
- Configuring subprograms. Import fields and fixed fields are defined, page separation is specified, an ODBC data source for the batch database is selected and an assignment of import fields to DMS object fields is performed.

Server Console

If an application server is started in a specific mode, you are provided with a console that offers you some information and configuration possibilities in order to monitor the server operations. The server console offers a number of information and configuration possibilities.

Transport of enaio® System Settings

Possibilities on how to transfer basic configurations between the enaio® test system and a productive system are described in the following.

A test system is recommended in particular for environments with workflow, events and complex data import processes. It allows you to develop and verify configurations before applying them to the productive system.

Structure of a Test System

For designing a test system you need a test license. Please contact OPTIMAL SYSTEMS or sales partners to obtain test licenses. Test licenses' validity is usually limited to five days and can be extended on demand.

Test and productive system should be identical to the greatest extend to enable meaningful tests and to simplify the configuration transport. Both configuration transfers between the enaio® test system and a productive system and vice versa are equivalent. That is why the test system setup is only briefly described in the following.

Installing a test system

The following requirements must be fulfilled to install a test system:

- Operating system

  enaio® server must run on the test and the productive system on the same operating system with the same language setting.
- Database

  Test and productive system should be using the same database system.
- Installation paths

  enaio® and all components should be installed in an identical path structure relative to the host.
- enaio® version

  Test and productive system should be using the same version.
- Service name

  Test and productive system should be using the same service name. This makes the transfer of registry settings easier. The service name is specified when requesting a test license.

The installation of enaio® is documented in the installation handbook.

Configuring the test system

If your productive system is already in use, the following basic data can be exported from the productive system and imported into the test system:

- Object definition (see Object Definition)

  The object definition is exported using enaio® editor. Import the export file into the test system and adapt the database tables using enaio® editor, too.
- User data and access rights (see User Data and Access Rights)

User data and access rights for archive objects are exported from the productive system with the automatic action 'User and group export'. Then import these data into user administration with the 'User and Group Import' action.

- W-Templates (see W-Templates)

  W-Templates are exported and imported using enaio® administrator.

Further basic system settings can be edited using enaio® enterprise-manager and enaio® administrator.

enaio® enterprise-manager (see Registry Entries) writes the configuration data into the registry and the database. All data which are administered in the database cannot be migrated. E.g. media administration data and license settings are stored in the database.

Registry data can be exported and imported using the registry editor. These data usually contain paths which should be verified.

enaio® administrator (see Configuration Entries in as.cfg) writes the configuration data into the registry and the configuration file `as.cfg`. This configuration file can be migrated if test system and productive system are mostly identical.

Object Definition

enaio® editor is used to define ECM objects – folder, register, and document types. Names and internal names of the objects and object elements are also specified. Objects and object elements are automatically assigned with a unique OS-GUID. When transferring the object definition this OS-GUID must also be migrated. The OS-GUID is not displayed in enaio® editor and can only be found in the object definition file or the database.

If you open an object definition file in enaio® editor and transfer objects, the OS-GUID is not applied from the object definition file, but a new GUID is assigned. Since some functions refer to the OS-GUID, make sure that the OS-GUIDs of the objects and object elements are identical in the productive and the test system.

That is only the case if you save the object definition in a file in enaio® editor and import this file into the target system. The import transfers all files from the object definition file into the database. Existing object definition data will be overwritten.

Some data are not saved in the object definition and must be verified:

- Structure-tree catalog

  The object definition contains path and name of the structure tree for the structure tree catalog. The structure tree file must be copied to the destination directory and the path must be adapted for the destination system using enaio® editor. To use it with enaio® webclient you will need to point to a copy of the structure tree file on the Web server in the configuration file.

- Address AddOn

  The configuration for the address AddOn is stored in a configuration file in the `etc` directory. The file must then be copied to the respective directory of the destination system. It is named according to the following schema:

  ```
  axadaddress_'OS-GUID'.xml
  ```

- Catalog AddOn

  The catalog AddOn configuration is saved into the `etc` folder of the data directory and must be copied into the respective directory of the destination system:

  ```
  axaddxmltree_'OS-GUID'.xml
  ```

- VB Script AddOn

  The object definition contains path and name of the script for the VB script AddOn. The script must be copied to the destination directory and the path must be adapted for the destination system using enaio® editor. The script may contain context-related information which must be verified.
- Application AddOn

  The object definition contains path and name of the application. Verify, if the application can be accessed this way on the destination system.
- Conversion AddOn

  The conversion AddOn uses a standard conversion factor. If the factor was changed in the corresponding `as.cfg` entry, this changed must also be made for the destination system.
- User AddOn and rights group AddOn

  Additional configurations for these AddOns can be applied with the corresponding `as.cfg` entries.

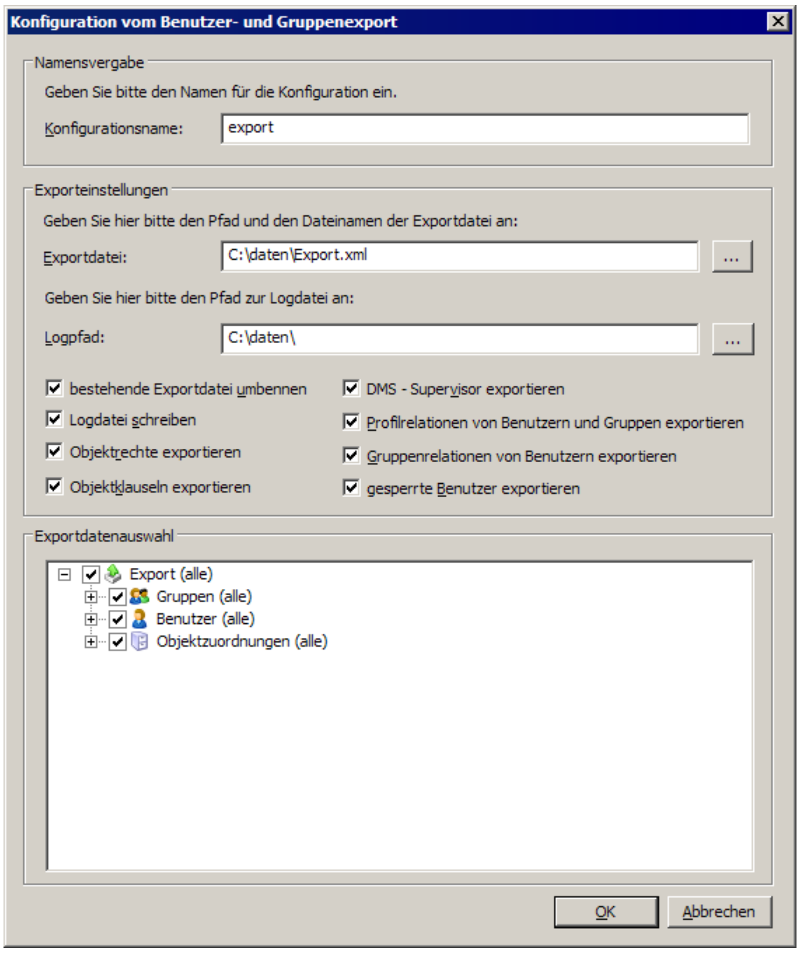
Images which are embedded in a form and user-specific icons for objects or list catalogs are saved in the object definition file and imported as well.

User Data and Access Rights

User data, group data and user-specific access rights can be exported and imported.

Exporting and importing group and user data is not possible if you have created areas for the remote user administration.

Export and import are executed as automatic actions in enaio® administrator.



Konfiguration vom Benutzer- und Gruppenexport

Namensvergabe

Geben Sie bitte den Namen für die Konfiguration ein.

Konfigurationsname: export

Exporteinstellungen

Geben Sie hier bitte den Pfad und den Dateinamen der Exportdatei an:

Exportdatei: C:\daten\Export.xml

Geben Sie hier bitte den Pfad zur Logdatei an:

Logpfad: C:\daten\

- ✓ bestehende Exportdatei umbennen
- ✓ Logdatei schreiben
- ✓ Objektrechte exportieren
- ✓ Objektklauseln exportieren
- ✓ DMS - Supervisor exportieren
- ✓ Profilrelationen von Benutzern und Gruppen exportieren
- ✓ Gruppenrelationen von Benutzern exportieren
- ✓ gesperrte Benutzer exportieren

Exportdatenauswahl

- ✓ Export (alle)
  - ✓ Gruppen (alle)
  - ✓ Benutzer (alle)
  - ✓ Objektzuordnungen (alle)

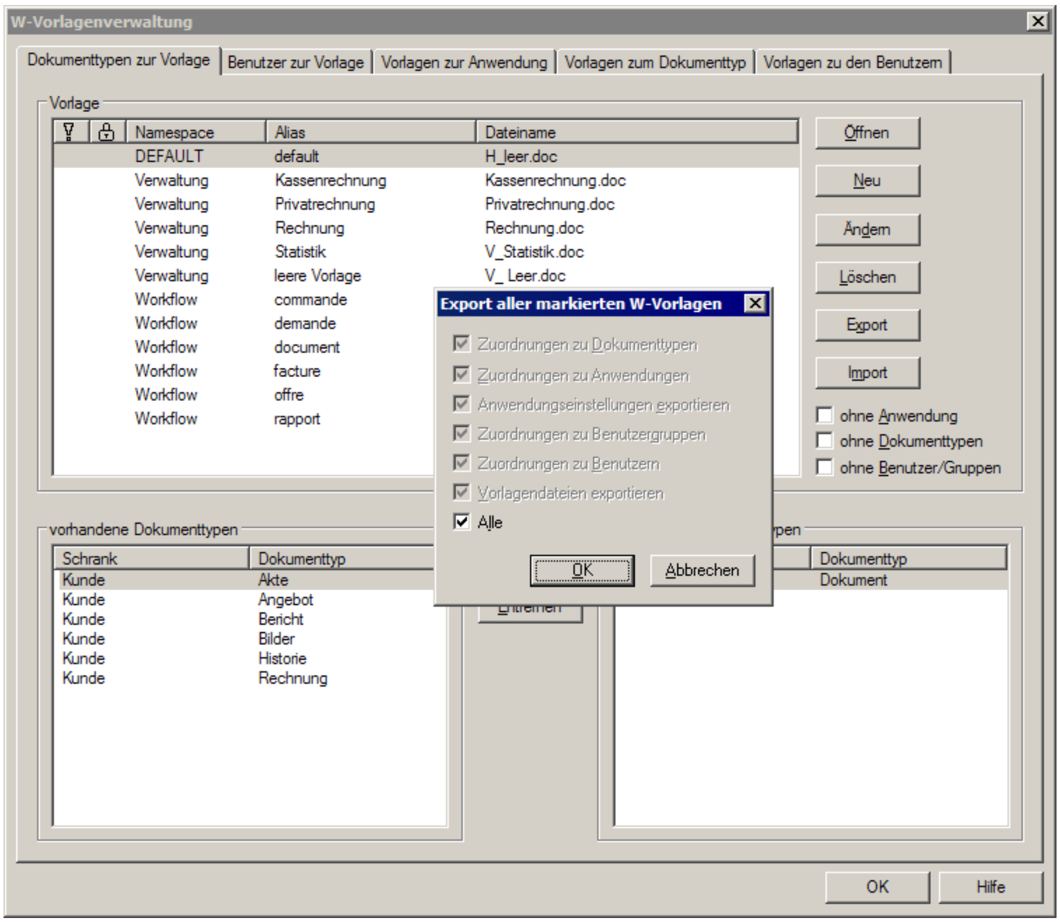
Data are exported with the action 'User and group export'. All data should be included during configuration. The exported data is imported into the destination system with the action 'User and group import'. All data should again be included during configuration.

These actions are documented in the enaio® administrator handbook.

W-Templates

In enaio® administrator template files are assigned to document types. This assignment is integrated with the rights system.

The assignments as well as the template files can be exported and imported via the W-template administration in enaio® administrator.


W-Vorlagenverwaltung

Dokumenttypen zur Vorlage | Benutzer zur Vorlage | Vorlagen zur Anwendung | Vorlagen zum Dokumenttyp | Vorlagen zu den Benutzern

Vorlage

Namespace	Alias	Dateiname
DEFAULT	default	H_leer.doc
Verwaltung	Kassenrechnung	Kassenrechnung.doc
Verwaltung	Privatrechnung	Privatrechnung.doc
Verwaltung	Rechnung	Rechnung.doc
Verwaltung	Statistik	V_Statistik.doc
Verwaltung	leere Vorlage	V_ Leer.doc
Workflow	commande	
Workflow	demande	
Workflow	document	
Workflow	facture	
Workflow	offre	
Workflow	rapport	

Öffnen | Neu | Ändern | Löschen | Export | Import

☐ ohne Anwendung
☐ ohne Dokumenttypen
☐ ohne Benutzer/Gruppen

Export aller markierten W-Vorlagen

☑ Zuordnungen zu Dokumenttypen
☑ Zuordnungen zu Anwendungen
☑ Anwendungseinstellungen exportieren
☑ Zuordnungen zu Benutzergruppen
☑ Zuordnungen zu Benutzern
☑ Vorlagendateien exportieren
☑ Alle

OK | Abbrechen

vorhandene Dokumenttypen

Schrank	Dokumenttyp
Kunde	Akte
Kunde	Angebot
Kunde	Bericht
Kunde	Bilder
Kunde	Historie
Kunde	Rechnung

Dokumenttyp
Dokument

OK | Hilfe

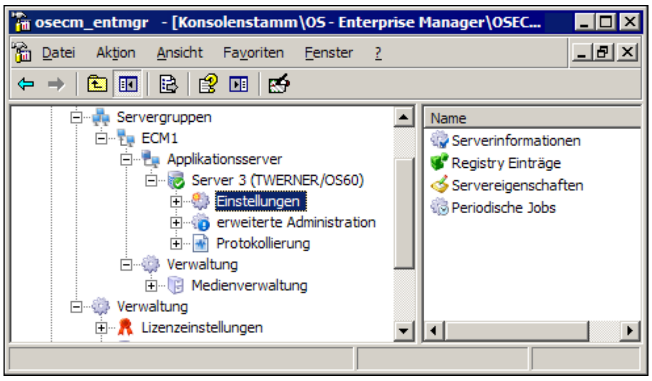
All data should be included during export and import.

Assigned template files are copied during export into the specified directory and assignment data is written into an XML file. During import template files are copied into the directory `etc\Templates` and assignment data is transferred to the database.

Registry Entries

enaio® enterprise-manager and enaio® administrator save enaio® server configuration data in the registry of the server computer.

enaio® enterprise-manager saves the data of the 'Settings' section in the registry.



Media administration data and license settings are stored in the database. These data cannot be migrated.

The data of the 'Settings' section can be exported and imported on the destination system using the registry editor.

The following structure must be exported:

`HKEY_LOCAL_MACHINE\SOFTWARE\OPTIMAL SYSTEMS\<service name>\`

If productive system and test system use the same service name, exported registry data can be directly imported into the destination system. If the service names are not identical, the service name must be changed in the registry file accordingly.

Registry data can contain a lot of context-related data, e.g. directory paths and URLs. These paths should be verified on the destination system after the import using enaio® enterprise-manager.

All registry data which are edited in enaio® administrator can also be modified with enaio® enterprise-manager; for that reason they will be transferred when migrating parts of the structure.

Configuration Entries in `as.cfg`

Configurations edited with enaio® administrator are saved in the registry, in the database or in the file `as.cfg` in the `etc` folder.

enaio® administrator saves almost all settings for the entire system which are configured on the tabs GENERAL ENAIO® CONFIGURATION into `as.cfg`.

The data in the file `as.cfg` are related to data which are managed in different locations. The following configuration data should be verified before transferring this file:

- Service name and port

  Service name and port are assigned to the parameter `comsting`. These data must be adapted if they are not identical.
- Automatic actions

  Included libraries for automatic actions on the ADDITIONS tab, default users including encrypted password, and configuration names of the automatic actions. The configuration data are saved in the database and cannot be migrated.

  If the file contains configurations which do not exist in the database, the automatic action must be configured manually. If the database contains data of configurations which are not specified in the file, this file cannot be accessed.

  The configuration data of the automatic actions for export and import of data can be exported and imported.
- Print Labeling

  Configuration of print labeling on the PRINT LABELING tab.
- Archive Print

  File format, path and background image names for archive print with enaio® printer. The images must be transferred to the destination system, if required.
- Web directory

  Configurations on the WEB DIRECTORY tab.
- Notes

  Configurations on the NOTES tab.
- Event administration

  Configurations on the EVENTS tab.
- Background image for enaio® client

  Integrates a background image using the START tab. enaio® administrator copies the image file to the `etc` folder of the data directory and names it `background.bmp`. The image must be transferred to the destination system, if required.

- E-mail administration

  Configuration of e-mail transfer from an integrated e-mail system (see chapter 'E-mail administration' in the 'enaio® administration' handbook).
- Checking for identical documents

  The check for identical documents is enabled with an entry in the file `as.cfg`.
- Database

  All configurations on the DATABASE tab.
- Documents

  All configurations on the DOCUMENTS tab.
- automatic update

  Automatic updates are enabled on the START tab, with other entries in the file `as.cfg` automatic updates can be limited to specified users.
- Deleting Variants

  The behavior when deleting variants is determined by an entry in the file `as.cfg`.
- Style sheets for XML document types

  Assignment of style sheets to XML document types. The style sheets must be transferred to the destination system, if required.
- Signature method

  The signature method is determined by an entry in the file `as.cfg`.

  The configurations for electronic signatures are saved by enaio® administrator in the database and must be again configured for the destination system.

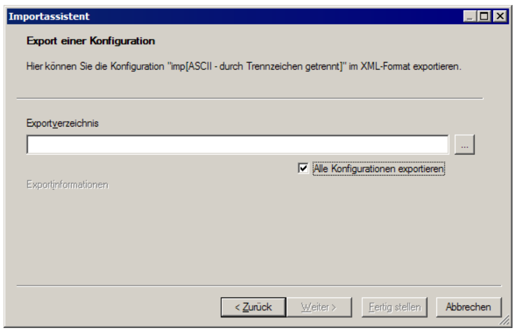
This list is incomplete. Further data for optional components are saved in the file `as.cfg`.

Automatic Actions for Data Import and Data Export

Import and export configurations of the automatic actions can be exported and imported on the destination system.

Automatic actions must be integrated on each system using enaio® administrator, ADDITIONS tab. Edit an existing import or export configuration or add an action via the AUTOMATIC ACTIONS dialog.

In the first configuration step, select the function EXPORT CONFIGURATION TO XML FILE, and specify the directory the configuration data should be exported into and select EXPORT ALL CONFIGURATIONS. For each configuration a file is created and saved into the directory.


Importassistent

Export einer Konfiguration

Hier können Sie die Konfiguration "imp[ASCII - durch Trennzeichen getrennt]" im XML-Format exportieren.

Exportverzeichnis

☑ Alle Konfigurationen exportieren

Exportinformationen

These files can then be imported into the destination system.

Edit an existing import or export configuration or add an action on the destination system.

First, select the option IMPORT CONFIGURATION FROM XML FILE OR FROM ASIMPEXP.CFG and specify a file. Only one file can be specified. The configuration is applied from the file and can be edited.

If a configuration with the same name already exists, it must be deleted prior to the import.

enaio® capture and COLD imports

Enter the DWORD value `ConfigExportCaptureCOLD` with the value `1` in the user-specific key `HKEY_CURRENT_USER\Software\Optimal Systems\Settings` in order to export the configuration data together with all configurations created over the import wizard for enaio® capture and COLD imports. After importing these data into the destination system, they will be available for the configuration of imports equally named in enaio® capture and for the automatic action 'COLD import'.

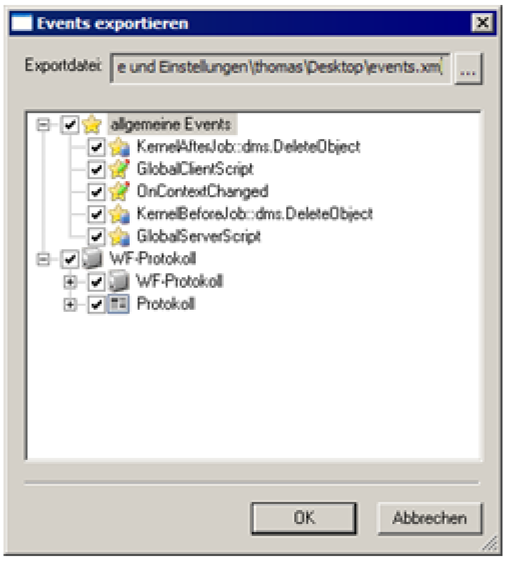
Please note that plausibility is not checked.

Events

Use enaio® client to export events from a system and import them into the destination system.

Select a user name in the object search area and choose EVENT EXPORT from the context menu.

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


Events exportieren

Exportdatei: e und Einstellungen\thomas\Desktop\events.xml

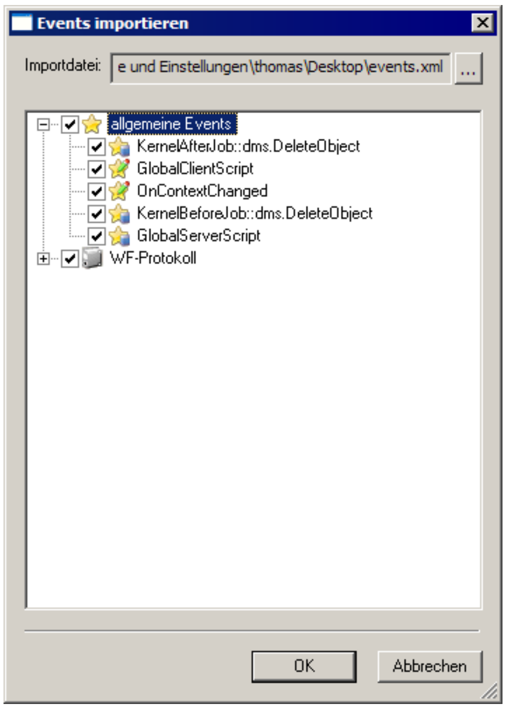
☑ allgemeine Events
- ☑ KernelAfterJob::dms.DeleteObject
- ☑ GlobalClientScript
- ☑ OnContextChanged
- ☑ KernelBeforeJob::dms.DeleteObject
- ☑ GlobalServerScript

☑ WF-Protokoll
- ☑ WF-Protokoll
- ☑ Protokoll

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

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

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



Individual scripts can also be exported and imported using the event editor.

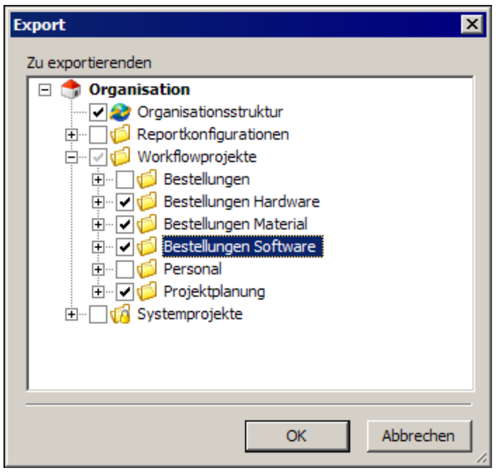
Workflow

Workflow process data consist of organizational data and model data. Both relate to each other. To transfer workflow processes, export the data of the organizational structure as well as the models.

Select the intended organization in the workspace of enaio® editor for workflow and choose **EXPORT ORGANIZATION** from the **FILE** menu.

Select the intended data in the export dialog.

Specify path and name for the export file.



This file can then be imported into the destination system.

enaio® webclient / Web Applications

enaio® webclient is configured using the administration console. The configuration data are saved in the file `osweb.properties` in the directory `\webapps\osweb\WEB-INF\conf\`.

This configuration file can be transferred to the destination system. This file contains the enaio® server connection data. These parameters must be adapted.

Configuration files of other enaio® components which are integrated as web applications can be transferred in the same way: enaio® documentviewer, enaio® contentfeeder und enaio® feedreader, enaio® fulltext.

The data also contain connection parameters which must be adapted.

Logging

With enaio® logging, extensive logging can be individually set up for every OS component, every workstation, and every user. The configurations are saved in a configuration file with the name `oxrpt.cfg` in the respective working directory.

This configuration file can be transferred to the respective directory on the destination system. If required, customize the log paths.

SQL Queries

The data of SQL queries are saved in the database and cannot be migrated. SQL queries thus must be created anew on the destination system. The SQL command, the configuration of the header, and the scripts can be copied and transferred into the configuration dialog of the destination system.

TAPI Integration

The configuration of the TAPI integration is saved in the configuration file `axcstapi-config.xml`. This configuration file can be transferred to the destination system.

enaio® document-storage

The configuration for enaio® document-storage is saved to the file `axvbdocstorage.xml` in the `etc` directory. This configuration file can be transferred to the destination system.

enaio® capture

Configurations for enaio® capture are saved in the database as well as in the configuration file `axindex.cfg` located in the directory `\clients\axindex` and in the configuration file `asform.cfg` located in the `etc` folder of the data directory. These configurations cannot be migrated.

enaio® portable

Configurations for the data export over enaio® portable are saved in the database and cannot be migrated.

Logging

Introduction to Logging

With enaio® logging, extensive logging can be individually set up for every OS component.

At least one action ID is assigned to every message from a component. This action ID allows log messages for flow, SQL, error and general log messages to be directed to different output channels and saved in different files.

A channel accepts messages with the same Action ID from one or more components and can store the messages in the internal log format, or in OXMISC format. The log level can be set individually for each channel.

The detailed channel properties settings allow you to configure workstation and component specific logging.

These settings are configured using enaio® manager for log files (`axprotocolcfg.exe`). The configuration is saved in a configuration file named `oxrpt.cfg`. Every component logs according to the logging settings in the configuration file in the application directory.

Using periodic jobs, channels can be enabled or disabled and the level changed.

OS_protocol-viewer opens logs in an internal log format itemizing the log very sophisticatedly. For example, messages are sorted according to user, session, thread, and computer. Filters and search features enable quick message localizing.

Additionally to this logging, further logs are created:

- enaio® server automatically writes log files at program launch and exit.
- In enaio® enterprise-manager you can enable access logging. All data accesses and queries are then logged.
- Additional logs can be enabled for the archiving process.
- Some automatic actions log additionally to the log settings.

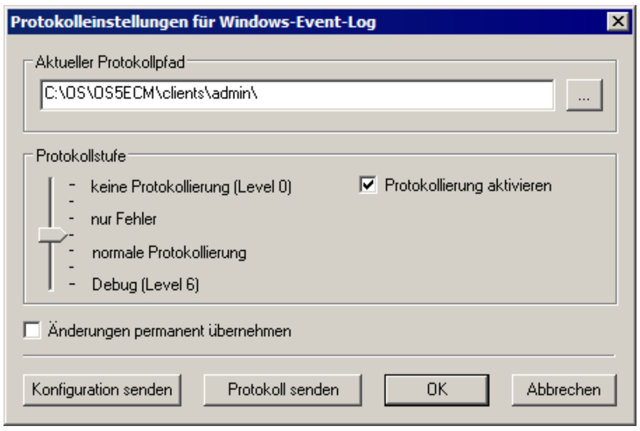
When installing enaio®, a logging directory is specified for the clients and for the server. This path is entered into the respective configuration file.

The server configuration file contains four predefined channels for flow, SQL, error, and log messages with the following names:

```
flwddmmyy.evn
sqlddmmyy.evn
logddmmyy.evn
errddmmyy.evn
```

In the client configuration file the channel for flow messages is predefined.

You can access the channel settings through the client:


Protokolleinstellungen für Windows-Event-Log

Aktueller Protokollpfad: C:\OS\OS5ECM\clients\admin\

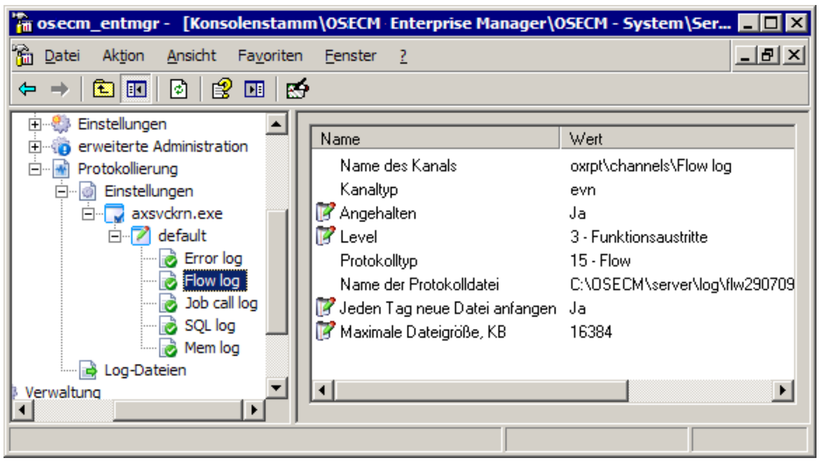
Protokollstufe
- keine Protokollierung (Level 0)
- nur Fehler
- normale Protokollierung
- Debug (Level 6)

[✓] Protokollierung aktivieren

[ ] Änderungen permanent übernehmen

Each user can change the data for the running program execution. Users with the system role 'Configuration complete system' can permanently adopt the settings.

Use enaio® enterprise-manager to modify the logging settings while enaio® server is running.


osecm_entmgr - [Konsolenstamm\OSECM Enterprise Manager\OSECM - System\Ser...

Name	Wert
Name des Kanals	oxrpt\channels\Flow log
Kanaltyp	evn
Angehalten	Ja
Level	3 - Funktionsaustritte
Protokolltyp	15 - Flow
Name der Protokolldatei	C:\OSECM\server\log\flw290709
Jeden Tag neue Datei anfangen	Ja
Maximale Dateigröße, KB	16384

enaio® manager-for-log-files

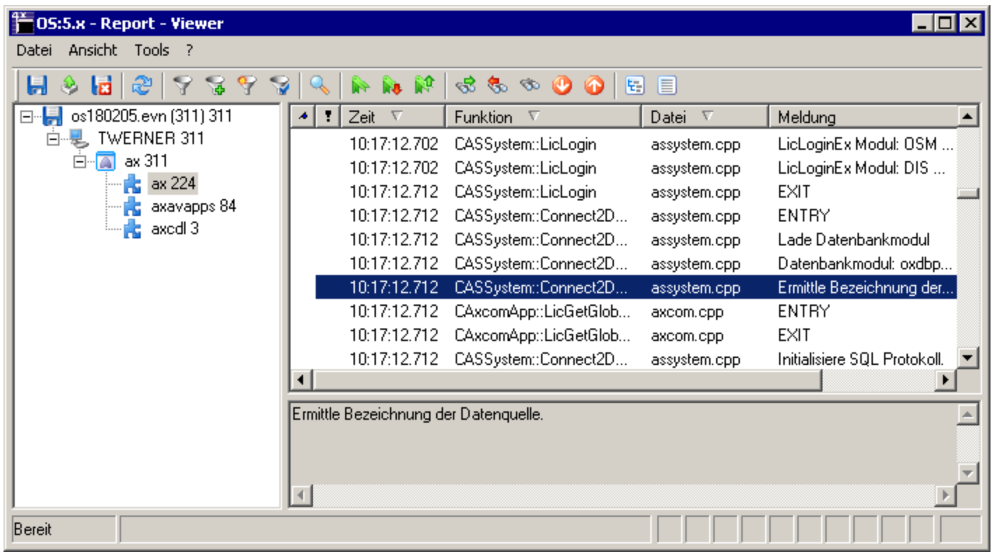
During installation of enaio®, the logging library `oxrpt.dll`, the configuration file `oxrpt.cfg` and the enaio® manager-for-logfiles application `axprotocolcfg.exe`, are installed in every directory containing OS components.

With enaio® manager-for-logfiles you can open the configuration file and edit the settings.

Each component uses the logging settings from the configuration file in the program directory when it is launched.

enaio® protocol-viewer

Logs stored in the internal log format (*.evn) can be opened with enaio® protocol-viewer. The program `axrptview.exe` can be found in every directory that contains enaio® components.



os180205.evn (311) 311
TWERNER 311
ax 311
ax 224
axavapps 84
axcdl 3

Zeit	Funktion	Datei	Meldung
10:17:12.702	CASSystem::LicLogin	assystem.cpp	LicLoginEx Modul: OSM ...
10:17:12.702	CASSystem::LicLogin	assystem.cpp	LicLoginEx Modul: DIS ...
10:17:12.712	CASSystem::LicLogin	assystem.cpp	EXIT
10:17:12.712	CASSystem::Connect2D...	assystem.cpp	ENTRY
10:17:12.712	CASSystem::Connect2D...	assystem.cpp	Lade Datenbankmodul
10:17:12.712	CASSystem::Connect2D...	assystem.cpp	Datenbankmodul: oxdbp...
10:17:12.712	CASSystem::Connect2D...	assystem.cpp	Ermittle Bezeichnung der...
10:17:12.712	CAxcomApp::LicGetGlob...	axcom.cpp	ENTRY
10:17:12.712	CAxcomApp::LicGetGlob...	axcom.cpp	EXIT
10:17:12.712	CASSystem::Connect2D...	assystem.cpp	Initialisiere SQL Protokoll.

Ermittle Bezeichnung der Datenquelle.

Different settings are available for the navigation pane on the left and the result pane on the right to clearly arrange different messages.

COM Interface

Overview

enaio® provides an interface to communicate with the client. This interface is available 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 chapter the term 'COM' is used exclusively.

The COM interface of enaio® client has more than one field of application. On the one hand, it allows you to access all DMS objects of enaio® client out of other applications. This allows you to start extended queries returning object hit lists, for example. It is also possible to create new objects in the enaio® system. On the other hand, you can use the COM interface to control enaio® client to a limited extend. For example, you can close windows or launch enaio® client. However, the control possibilities for applications are less extensive than for MS applications.

More information can be found in the 'COM interface of enaio® client' manual.

Interfaces of the COM Interface

The COM interface offers its interface over the type library `ax.tlb`.

The following interfaces are available:

- Application. This is the most extensive interface with numerous functions
- Application2. This interface provides additional functions for application control and for extended information.
- Window. This interface provides functions to control individual windows in enaio® client.
- Windows. This interface provides three functions to control the collection of all enaio® client windows.

The interface which provides the most functions is 'Application'. The COM functions of the application interface can be divided into the following function areas.

- Object-related commands,
- Inserting, updating and deleting objects,
- document processing,
- application control,
- information,
- conversion and file creation.

An assignment of COM functions to the corresponding functional area is discussed in the section 'Description of the COM interfaces' below.

COM Interface and the enaio® Security System

The security system controls the access to DMS objects of enaio®. The COM interface can accept or ignore these object rights. The COM interface behavior can be set with the property 'Mind security system' for each object type in enaio® editor.

The COM interface can address all object types the logged in user can see in enaio® client. In addition, it addresses all objects of which the property 'Mind security system' is set 'No'. For objects that can be seen by the logged in user and that do not mind the security system, the right limitations set for the logged in user will be ignored.

When setting the value 'No' to the property 'Mind security system', users can, for example, determine object values with the data transfer macros of MS Word even if they usually have no access to these objects. However, they need to know the field names of objects. Finding out these field names needs your programming knowledge. For that reason, the COM interface should mind the security system for respective object types.

enaio® capture

Overview

enaio® capture enables capture, indexing, validation, and import of large numbers of records. Thereby, process steps are separated and automated as far as possible. Each step processes records of different operations. This method can be applied in processes including frequent capture of a large number of paper records.

A detailed description can be found in the enaio® capture handbook.

The functions of enaio® capture can be separated in four basic steps:

- Capturing document files (Scanning)

  Here, data are scanned and imported. High performance scanners are used for scanning. The data import transfers numerous files that have usually been created frequently through other DV systems.
- Character recognition

  Captured document images are processed through OCR engines. Texts are recognized from present image data and used later for indexing.
- Validation

  Captured document images and recognized index data are checked visually by the processor. Thereby, automatic validation through VB scripts supports the processor.
- Import

  In the last step, DMS objects are imported into the enaio® system. Usually, this step is realized through a component also applied for automatic import actions. In addition, dBase files can also be created for import into other system and it is possible to automatically send created enaio® objects by e-mail to enaio® users.

There are different process flows available for different record types. These process flows are called configurations. Configurations consists of various process steps, each processed through a subprogram. Paper records are processed in batches. As soon as a record has been scanned, further process steps use images of these records only. Paper records can be archived and destroyed after they have been captured and saved digitally if allowed by legal regulations.

Productive application of enaio® capture is prepared through administrative setup operations. Initially, designated workstations must be set up. After having fulfilled the conditions for the work with enaio® capture, e.g. setup of necessary hardware and software, scan stations are equipped with required client components over network installation. Then, enaio® capture must be configured. As a result, enaio® capture is available and can be applied productively.

Administration of enaio® capture

This section describes the preparation of enaio® capture workstations, the installation, configuration, and operation of enaio® capture. After installation, you need to define the configurations and subprograms. A configuration consists of subprograms that process and forward data. Subprograms can be executed on different workstations and automatic execution can be set up, as well.

Two configuration files control the process flow of a batch. Data generated during processing are cached in an ODBC database (MS Access recommended).

Requirements for Using enaio® capture

Technical requirements

enaio® capture requires additional hardware and software, not part of the enaio® system:

- Connective adapters for scanners: scanners are addressed over Kofax and Twain so Kofax and SCSI cards can be used. Furthermore, USB ports are supported.
- Scanner hardware and drivers: supported scanners must be used that can manage the number of documents. Kofax and TWAIN drivers are supported.
- OCR engine: FineReader 8, 7 and Kofax engines are supported. FineReader possibly requires dongles.
- Kofax retrieval engine: install the Kofax retrieval engine for scan subprograms on computers without Kofax scan engine. It enables the display of image contents.
- E-mail clients: enaio® capture is able to send captured images or references to generated DMS objects by e-mail. For this purpose, a MAPI client, e.g. MS Outlook, must be installed.

Organizational requirements

In enaio® capture single process steps of document capture are separated in such a way that as many steps as possible can be processed automatically, such as record scanning, barcode recognition (with unique folder ID), and document import. For processing, a larger number of similar documents is collected in batches. Record processing with enaio® capture is integrated in the organizational environment.

Authorizations: You must define users in enaio® administrator that are allowed to

- configure enaio® capture
- start enaio® capture

The enaio® setup installs enaio® capture on the file server. Network installations are run on workstations whereby they are connected to the installed enaio® capture files.

Otherwise, you can locally install the client. With this installation type the actions of different enaio® capture stations cannot be aligned until you have changed the local client installation into a file server installation through shared folders. Several enaio® capture stations can be aligned if all stations can access two configuration files and the batch database.

Following conditions have to be fulfilled for a successful installation:

- Hardware and software requirements are fulfilled.
- A running enaio® server is available.

- enaio® capture can be started with a valid license only.

enaio® capture and the subprogram types offer an automatic update option. In case there is a more up to date version in the enaio® capture directory `etc\update\index`, all components are updated automatically if the update option has been configured in enaio® administrator.

Standard subprograms

This table provides an overview of the installed enaio® capture components. Programs are arranged based on their function.

Control	Config.	Scanning	OCR/DB	Validation	Import
AXINDEX	AXRCCONF	AXICSCAN		AXVALID	AXIMPORT
AXIASTOP		AXDSCAN	AXICSVR	SCRIPTE	AXMAILDC*
		AXTWSCAN	AXNOOCR		AXIMPMDC*
		AXIMGIMP	AXFINER7		AXPARTDC
					AXSTARTWF

Table 47: enaio® capture subprograms

Subprograms flagged with an asterisk '*' do configure themselves, i.e. they do not use the component AXRCCONF. For configuration, scanning, OCR/DB, validation, and import you can apply individual programs provided that the interfaces of enaio® capture are used correctly. For more information please contact OPTIMAL SYSTEMS.

subprograms are briefly described here. They are grouped according to the following basic functions: image input, basic indexing, validation, and output.

Control

- axindex

  The corresponding executable file is `axindex.exe`. It can start without parameters. Depending on the logged in user, enaio® capture functions are then available. If you start the file with the start parameter `/auto`, enaio® capture will start in auto mode that does neither allow configuration nor manual batch processing. Subprograms automatically process batches if configured respectively.

- AXIASTOP

  The corresponding executable file is `axiastop.exe`. It starts automatically if enaio® capture runs in the automatic mode. As many batches to be processed quickly one after another in automatic mode make exiting enaio® capture more complicated, AXIASTOP allows you to easily exit enaio® capture.

Configuration

The following programs contain configuration functions for subprograms:

- axrcconf

  The corresponding executable file is `axrcconf.exe`. It is used for configuration of most of the standard subprograms. The following settings are made therein:

  - Field definition

  Define fields in which values captured during batch processing are saved.

  - Page separation

  Configure the criterion deciding when a new document begins in a document batch.

- Import

  Select the ODBC source to the batch database and assign the batch fields to import fields in enaio®.

- Axmaildc and aximpmdc

  Both programs configure themselves. At setup, you have to specify the location of a prepared configuration file only.

- Others

  Specify individual subprograms for subprograms not contained in the standard delivery package.

Scan

These programs create and import image files into enaio® capture. The following subprograms come with the standard enaio® package:

- axicscan

  The corresponding executable file is `axicscan.exe`. It enables scanning and integrates barcode recognition with Kofax engines. Upon completion of a scanning process and forwarding a batch, `axicscan` creates a new table in the batch database. That is why this subprogram is added to the function `OCR/DB`.

- axdscan

  The corresponding executable file is `axdscan.exe`. It enables scanning with Kofax engines. `axdscan` does not create a new table in the batch database.

- atwscan

  The corresponding executable file is `atwscan.exe`. It enables scanning with TWAIN scanners. `atwscan` does not create a new table in the batch database.

- aximgimp

  The corresponding executable file is `aximgimp.exe`. It enables image file import into the process flow of enaio® capture. `aximpgimp` does not create a new table in the batch database.

OCR/DB

In the following subprograms are described which create a table in the batch database and insert indexing data captured by character recognition into the table.

- axicsrv

  The corresponding executable file is `axicsrv.exe`. It is used for barcode recognition with Kofax engines. Upon completion of a scanning process and forwarding a batch, axicsrv creates a new table in the batch database.

- axnoocr

  The corresponding executable file is `axnoocr.exe`. It does not perform any character recognition and creates a new, but empty table in the batch database.

- axfiner7

  The corresponding executable file is `axfiner7.exe`. It is used for character recognition with FineReader Engines. Upon completion of a scanning process and forwarding a batch, axrecog creates a new table in the batch database.

Validation

These are the subprograms which can be used for data and image validation.

- axvalid

  The corresponding executable file is `axvalid.exe`. Using this program captured images and existing indexing data are checked by the editor and corrected, if necessary. Furthermore, axvalid is highly suitable for subsequent entering of indexing data. Another important feature is the possibility to validate captured values with a VB script.

Import

These are the subprograms which can be used to import data and document files into enaio® or to prepare them for export into other systems.

- aximport

  The corresponding executable file is `aximport.exe`. First, the batch is separated into single documents. Then the data of the batch database and the created image files are imported into the enaio® system which is connected to enaio® capture. This is based on the import engine which is also used by the automatic action 'Import/Export'.

- axmaildc

  The corresponding executable file is `axmaildc.exe`. The data of the batch database and the created image files are sent by e-mail to the configured recipients without being imported into the enaio® system. The recipient can then create enaio® objects manually.

- aximpmdc

  The corresponding executable file is axmaildc.exe. First, the batch is separated into single documents. Then the data of the batch database and the created image files are imported into the enaio® system which is connected to enaio® capture. So far the functionality is identical with aximport. Then references to the created objects are sent by e-mail to the configured recipients without being imported into the enaio® system. The recipient can then edit the created enaio® objects.

- axstartwf

  The corresponding executable file is `axstartwf.exe`. This subprogram imports the captured document into the workflow component of enaio®. No retrievable DMS objects are created. The defined batch fields can be assigned to workflow variables, objects fields are not assigned.

Individual subprograms

In addition to the listed subprograms, individual programs can be integrated into the processing. An interface defined by OS will be required. In order to customize the interface, please get in touch with the consulting department of OS.

Distribution across several Network Workstations

enaio® capture can be configured to allow batches to be processed on a single workstation. Working at one workstation is only useful for a limited number of documents. If the number of receipts exceed a certain amount, one person is no longer able to process everything using one workstation. Consequently, work must be distributed across several workstations.

As some of the subprograms can be executed automatically, e.g. character recognition and import subprograms, and some always require user input, e.g.

scanning and validating, it is not useful to just duplicate workstations and to execute all processing steps on every workstation. A functionally structured processing makes more sense. Both processing types, automated processing steps and processing steps which require user input, should be separated.

The work process may look as follows:

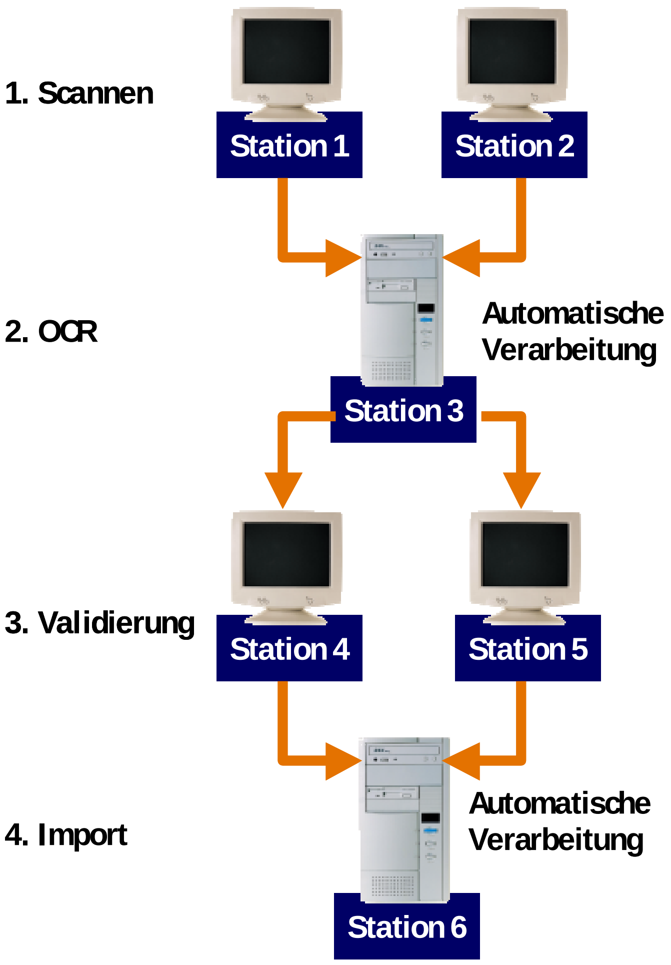


Figure 60: Distributed processing

Only the progress of the processing of the individual batches is illustrated, the paper input is not illustrated. Please note that two programs are executed on station 3, OCR and IMPORT.

To set up the illustrated process, the following configuration steps are required:

- The licensing of the individual stations in enaio® administrator
- User rights in enaio® administrator
- enaio® capture rights for individual configurations

Licensing

Licenses are set up in enaio® administrator. The following steps are required for licensing:

- Enter each station
- Individual license modules can then be added to each station

The following licenses must be assigned to use the respective properties:

module	Application
AIX	enaio® capture

SCA	Scanning components
STW	TWAIN Scanning
SIC	Scanning and recognition (Kofax)
REK	Recognition (Kofax)
RER	Recognition (FineReader)
W2D	W to D conversion (barcode)
SFI	Scan file import
VAL	Validation
AIE	Import
AMT	MAPI export
AVG	Start workflow

Table 48: License modules

Once the licenses have been assigned correctly, the applications can be executed on each station. This does not guarantee that the user can log on to enaio® capture. Hence, object and application permissions must be set up in enaio® administrator.

Detailed information on assigning licenses can be found in the enaio® administrator handbook.

Object and application permissions

Two things need to be considered regarding setting up permissions for enaio® capture users. The visibility of individual enaio® capture configurations is set up via user groups. The permissions to configure and execute enaio® capture can be set for each user, though.

To structure the visibility of enaio® capture configurations create different user groups. Then create users, assign the required application permissions and assign user groups to the users. Later in enaio® capture, set the groups for each configuration who are permitted to view the configuration.

Detailed information on the enaio® security system can be found in the enaio® administrator handbook. An enaio® capture administrator is permitted to see all configurations.

Additional information for processing across the network

This completes the setup of all stations on the network and allows them to perform their assigned function. User groups and users registered in enaio® are set up, too.

Also to be considered:

- If you want to distribute the load during batch processing across these stations, all stations must access a common batch directory. To do so only perform one central installation of enaio® capture. Create a shortcut to the central axindex.exe for each station.
- Depending on the installed modules, display, scan or recognition engines must be installed on each station. A scan engines must be installed for a scanning station and a designated display engine must be installed for the validation station. Station 3 in the example figure above requires a recognition engine.
- The batches are distributed automatically with the help of a control component based on the FIFO principle: the first incoming batch is

processed by the first requesting station. This is valid for the automatic processing. Each batch can be started manually.

Appendix

System Roles

This is a complete overview of all system roles which are available for the system. Different system roles are valid for different client applications and manage the possible action scope of the user logged on to this client.

The following system roles can be assigned to users individually in enaio® administrator:

DMS: Set up local security groups	Open configuration of remote areas.
DMS: Supervisor	Contains all other system roles (except Workflow and enaio® enterprise-manager)
Administrator:	
Start	Allows starting enaio® administrator.
Configuration complete system	Open the configuration dialog of the entire system (LDAP, adding automatic actions etc.).
Configuration security system	Open the configuration of the security system. Set up and configure users, groups and object rights.
Configuration Local security groups	Restricted configuration of the security system for remote areas.
Configuration W-templates	Allows opening the W-templates management to configure W-templates and associated applications.
Configuration of Archive Print	Open configuration of the archive print.
Start automatic actions	Start already set up configurations of automatic actions.
Configure automatic actions	Create and set up configurations for automatic actions.
View user tray	Allows viewing private filing trays of all users in enaio® administrator.
Change object rights	Allows changing the owner of an object. Either a different user can be assigned as the owner or the ownership can be taken by oneself (enaio® client).
Editor:	
Starting	Allows starting enaio® editor.
Edit object definition:	Change and complete the object

	definition. With this role only, a database cannot be altered.
Customizing the database	Database adaption can be performed.
Start:	
Starting	Allows starting enaio® start in order to execute automatic actions.
Capture:	
Starting	Enables to start enaio® capture.
Configure	Enables to set up configurations for enaio® capture.
Client:	
Starting	Enables to start enaio® client.
Save personal settings	Personal user settings cannot be set. It is not possible to save queries or documents in the archive area of enaio® client.
Customize the ribbon	
Open history	The editing history can be viewed if it was activated.
Recover data from history	Data of the editing history can be restored. (Requirement: the role 'Client: Open history' role is available.)
Configure history of individual objects	Allows the history to be configured for a single document, provided that this option is available in the used object definition. (Requirement: the role 'Client: Open history' role is available.)
Start expert mode	Enables the use of the expert mode to search for objects.
Export from hit list	Export index data and document files from a hit list.
Print documents	Print documents in enaio® client.
Open notes	Notes for objects are displayed.
Edit notes	Create and edit object notes.
Show personal trash can	Show personal trash can, from which deleted data can be restored.
View system trash can	Displays the system trash can in which deleted objects of all users are visible. (Requirement: the role 'Client: Show personal trash can' is available.)
Delete objects from trash can	Allows irreversible deletion of objects from the trash can. Objects cannot be restored anymore. (Requirement: the role 'Client: Show personal trash can' is available.)
Execute SQL queries	Allows executing extended queries.
Send content as e-mail	Objects can be sent by e-mail to external recipients who are not set up in the enaio® system.
Open properties	The properties dialog of an object can be

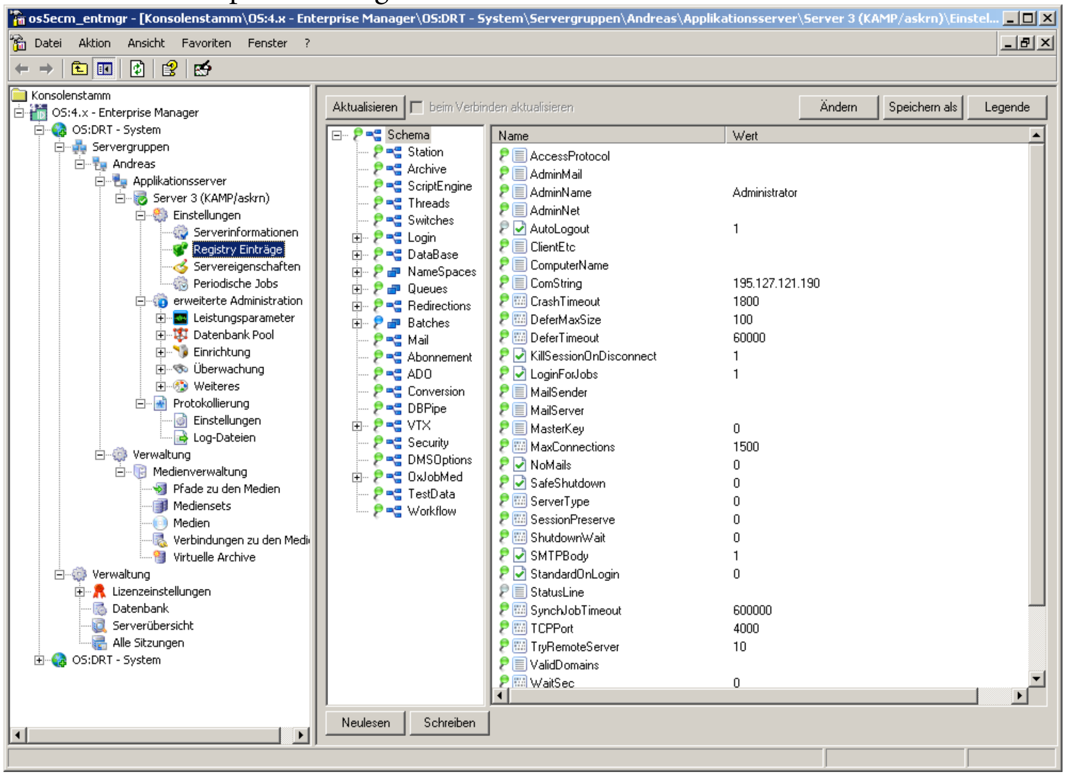
	displayed.
Move objects	Objects can be moved if this is permitted in enaio® administrator.
Move across cabinets	Allows documents to be moved across cabinets.
Use workflow	Using workflow is allowed for this user.
Workflow substitute configuration	Set up workflow substitute for one self.
Workflow process administration	Enables restricted administration of running processes in enaio® client.
Edit circulation slip workflow	Permits to modify the circulation slip of the workflow.
Workflow manage private circulation slip templates	Enables to manage and delete the templates of private circulation slips.
Workflow manage public circulation slip templates	Enables to manage and delete the templates of public circulation slips.
User Object search	Permits the use of the Object search.
Use Navigation	The user is permitted to use the navigation pane.
Run default search	If users should only perform searches using saved queries, the default search can be switched off.
Add icon	Adding icons to the system's data pool is permitted.
Delete icon	Deleting icons from the system's data pool is permitted.
Edit static layers of other users	Permits to edit static layers created by anybody else.
Edit SQL queries	SQL queries can be created and edited.
Administering Subscriptions	Administer and delete subscriptions of all users. Subscriptions can also be set up for any user.
Administration of follow-ups	Enables to administer and delete follow-ups of all users.
Debug events	Client events can be executed in debug mode.
Create events	Events for client and server can be edited.
Depict relations	The relation depiction/relation list can be opened. As a requirement, the system must be set up for relation links.
Create visualization	Creating a visualization of relations and permanent storage of this visualization in the corresponding document type.
Administer visualization	General configuration for relation visualization
Change archiving state	Users can give documents the property 'archivable' or 'not archivable.'
Delete archived documents	Delete documents that have already been

	archived by the system.
Execute collective editing	Permits changes in batch mode for more than one object selected in a hit list. (Note: This role may overwrite existing write-protected object expressions of the rights system.)
Always show variant administration	Users who have no write permissions for index data or objects can open the variant administration in read-only mode, but are not allowed to create new variants or assign the 'active variant' status to other variants.
Show favorites	For every user, a 'Favorites' portfolio is managed; its objects are accessible as favorites on mobile devices through OS|mobileDMS.
WebClient:	
Starting	Starts enaio® web client
WF-Admin:	
Starting	Start enaio® administrator-for-workflow.
WF-Editor:	
Starting	Start enaio® editor for workflow.
Edit organization	Edit a workflow organization.
Notify users as present/absent	The presence of any workflow user can be set.
Create model	Create a workflow model.
WF-Processes:	
Start via Import	Enables starting workflow processes through system import using the automatic action 'Data and document import'.
WF-Simulation:	
Starting	Start a workflow simulation.
WF-Script:	
Starting	Start a runtime for workflow scripts.
Enterprise-Manager:	
Starting	Allows starting enaio® enterprise-manager.
Server:	
Switch job context	

enaio® enterprise-manager

For the previous enaio® version, enaio® enterprise-manager has been developed to administrate server-specific settings and to configure several application servers. enaio® enterprise-manager is a snap-in for the Microsoft Management Console.

Through enaio® enterprise-manager, several settings have been centralized and are only made available in enaio® enterprise-manager. Furthermore, you can now access the database, view table contents, and manipulate the database using SQL with enaio® enterprise-manager.


os5ecm_entmgr - [Konsolenstamm\OS:4.x - Enterprise Manager\OS:DRT - System\Servergruppen\Andreas\Applikationsserver\Server 3 (KAMP/askrn)\Einstel...
Datei
Aktion
Ansicht
Favoriten
Fenster
?
Aktualisieren
beim Verbinden aktualisieren
Ändern
Speichern als
Legende
Neulesen
Schreiben

```
Konsolenstamm
  OS:4.x - Enterprise Manager
    OS:DRT - System
      Servergruppen
        Andreas
          Applikationsserver
            Server 3 (KAMP/askrn)
              Einstellungen
                Serverinformationen
                Registry Einträge
                Servereigenschaften
                Periodische Jobs
              erweiterte Administration
                Leistungsparameter
                Datenbank Pool
                Einrichtung
                Überwachung
                Weiteres
              Protokollierung
                Einstellungen
                Log-Dateien
          Verwaltung
            Medienverwaltung
              Pfade zu den Medien
              Mediensets
              Medien
              Verbindungen zu den Medi
              Virtuelle Archive
      Verwaltung
        Lizenzeinstellungen
        Datenbank
        Serverübersicht
        Alle Sitzungen
    OS:DRT - System
```

```
Schema
  Station
  Archive
  ScriptEngine
  Threads
  Switches
  Login
  DataBase
  NameSpaces
  Queues
  Redirections
  Batches
  Mail
  Abonnement
  ADO
  Conversion
  DBPipe
  VTX
  Security
  DMSOptions
  OxJobMed
  TestData
  Workflow
```

Name	Wert
AccessProtocol	
AdminMail	
AdminName	Administrator
AdminNet	
AutoLogout	1
ClientEtc	
ComputerName	
ComString	195.127.121.190
CrashTimeout	1800
DeferMaxSize	100
DeferTimeout	60000
KillSessionOnDisconnect	1
LoginForJobs	1
MailSender	
MailServer	
MasterKey	0
MaxConnections	1500
NoMails	0
SafeShutdown	0
ServerType	0
SessionPreserve	0
ShutdownWait	0
SMTPBody	1
StandardOnLogin	0
StatusLine	
SynchJobTimeout	600000
TCPPort	4000
TryRemoteServer	10
ValidDomains	
WaitSec	0

Figure 61

Use enaio® enterprise-manager to visualize the server infrastructure. This way, all server groups of a system are grouped in a server family. Within a server group, the set ups of all server instances can be determined. The following operations can also be performed with enaio® enterprise-manager:

Category	Area	operation	Description
Console root	enaio® enterprise -manager	Add/remove server families	Enables the registration of enaio® systems.
Server family	Server groups		Server groups are automatically retrieved from the database.
	Administration	License settings/licenses	Functions of the license editor used to assign modules to individual workstations.
	Administration	License settings/license monitor	Display of currently assigned licenses.
	Administration	Database/tables	View of all database tables.
		Database/SQL input	Possibility to manipulate data-
	Administration	All Sessions	View of all sessions which are entered in the database table.

	Administration	Server Overview	View of all installed servers and server groups.
	Administration	Prepared Jobs	View of all 'PreparedJobs' which are entered in the database table.
Server group	Administration	Path to Media	Administration of the table PATH.
	Administration	Media Sets	Administration of the table SETS.
	Administration	Media	Administration of the tables MEDIEN.
	Administration	Connections to Media	Administration of the table MEDREL.
	Administration	Virtual Archives	Administration of 3rd party archives.
Server group	Application Server		Server instances are retrieved from the database.
Application Server	Preferences	Server information	Determine default parameter for the server.
		Prepared Jobs	View of 'PreparedJobs' which can be processed by this server.
	Preferences	Registry Entries	Possibility to determine registry entries remotely.
	Preferences	Server Properties	Server properties settings.
	Preferences	Periodic jobs	Settings for periodically executed batches.
	Advanced Administration	Performance parameters/process information	Windows information on the process and the running threads.
		Performance parameters/job threads	Overview of all threads, which jobs were recently processed or are currently being processed.
		Performance parameters/loaded modules	View of all libraries which were loaded by the server process.
		Database pool piping	Overview of all database connections which are used for DBPIPING.

		Database pool/Read threads	Overview of all database connections which are used for Read Threads.
		Setup/Engines	All loaded executors with their workspaces and statistics on the job execution. Executors can be unloaded resp. reloaded here.
		Monitoring/Connections/Active clients	All clients which are currently active on the server with the possibility to administratively quit individual sessions.
		Monitoring/Connections/Connections to other servers	All active connections to other servers with the possibility to administratively quit/restore individual sessions.
		Monitoring/Job queues	Worker queues monitoring
		Monitoring/Notifications	Overview of notifications which enaio® enterprise-manager receives from the server. Furthermore, notifications can be sent to any client logged on to the server.
		Monitoring/Job calls	Monitoring of all jobs which the server receives from the clients.
		Monitoring/Prepared jobs	Overview of 'PreparedJobs' which can be processed by this server.
		Additional/Persistent memory	Overview of Start and Stop information which are stored by the server in the database. Information on the last 10 server starts are stored.

		Miscellaneous/Kernel events	Overview of internal events which can be resolved by the server. These events can be subscribed to by enaio® enterprise-manager.
	Logging	Preferences	Possibility to change the log settings during runtime (not permanently)
		Log files	Overview of the log files which are currently located in the server log directory. These log files can be loaded or deleted here.

Registry Entries for enaio® server

The registry entries for the current version of enaio® server are documented in a HTML file. You can create this HTML file for the currently installed server version by executing the Java script *oxentmgr_XML_to_HTML.js*.

This script file is located in the installation directory of enaio® administrator. The same script creates in the same directory the file *oxentmgr_sp.html* in which the available registry entries of the currently installed version of enaio® server are entered. All listed registry entries can be edited with enaio® enterprise-manager.

Starting enaio® server and the Command Line

Switches and Variables

The server has some internal switches and variables which can be used in the command line.

`Debug`: defines if a debugger is requested right after the analysis of the command line. Default value: OFF. If it is set to ON, the operating system is requested to start a debugger for the starting server process. This is useful for debugging the service. It is not advised to have this switch set to ON for normal operation. Default value: OFF

`Service`: if this switch is set to ON, the server communicates with the service control manager and attempts to start as a service. If this switch is set to OFF, the server will start as an exe. Default value: ON

`Console`: if this switch is set to ON, it is attempted at system start to start a console and to establish a connection. Default value: OFF

`Verbose`: if this switch is set to ON, the error which has prevented the system start is displayed in a message box. This depends on the server being started as a service or not. Default value: OFF

`ReportEnabled`: if this switch is set to ON, server startup and shutdown is reported. The default value is ON.

Created reports are saved to the `\server\ostemp\` directory.

They are named as follows:

`startup.txt` and `shutdown.txt`

The 'ReportDateTime' switch allows you to add the creation date to the name: `startup_YYYYMMDD_HHMMSS.txt` and `shutdown_YYYYMMDD_HHMMSS.txt`

`ReportRegistry`: if this switch is set to ON, also parameters are reported in the start report (if available) which the kernel has read from the registry. This requires space and makes it more difficult to read. Default value: OFF

`ReportLicense`: if this switch is set to ON, all modules are logged which are currently listed in oslicense, and all modules which are listed in the table osresources. Default value: OFF

`ReportFiles`: if this switch is set to ON, all system files are reported which are entered in the table osresources. Default value: OFF

`ReportDatetime`: if this switch is set to ON, the current time is used to name the report file. Default value: OFF

`Name`: this variable defines the name of the service and the registry key from which all settings can be read.

`Interval:` specifies time periods for single starting steps. A warning appears if the time period is exceeded. Several time periods can follow on each other.

Start Procedure

The file oxrpt.dll is loaded and initialized. In doing so, the alias 'axsvckrn' is used. If initialization fails, the process will be quit with the error code SRETCODE_REPORTINIT.

All switches and variables are initialized with default values. It is also noted that the command line has not yet been analyzed and that the server is unaware if the user wants to start it as a service or as a normal application. These settings are saved in the internal variable ServerSwitchDefined.

Then the command line is analyzed.

Then, if the debug switch is set to ON, a debugger is requested.

Then the name is verified. If it is blank, the default value 'askrn' is used.

After that perflib is loaded but not yet activated.

Then it is verified if a process with the same name is already active. To do so a system-wide flag is used having the name as part of its name. This means that it is verified if we are the first using this value as a name. If this verification fails a log is written, a message box displayed in Verbose mode, a Windows Event Log event created and the process quit with the error code SRETCODE_ALREADYRUNNING.

Afterwards, the variable ServerSwitchDefined is checked. If it has the value FALSE, the user has not defined the server switch and you need to find out, how to be started. All processes are listed with the help of the ToolHelp library and the parent process of our process is determined. In case it is 'service.exe' the server assumes that it has been started as service from the ServiceControlManager. The switch service is thus set to ON. If not, the switch is set to OFF.

When the switch service is set to OFF; the switch console is necessarily set to ON: if not as service, the server always starts with the console. In other words, if not as a service, always with the console. If as a service, either with or without the console according to the command line parameter.

If service is set to ON and the MainThread eventually starts, it always communicates with the SCM. The hereby occurring process must be described separately. Then, the last log entry is written.

Command Line

The command line has the following format (in BNF syntax):

```
Axsvrkrn.exe parameter-list
parameter-list ::= [ parameter [ blank parameter-list ] ]
blank ::= „ „ - space character, difficult to depict in Word
parameter ::= {'-' | '/' | '\'} {debug | action | console | service |
name | paramfile | verbose | report}
action ::= { install | uninstall }
install ::= { ‚I' | ‚i' } - sets the value of the mode switch to INSTALL
uninstall ::= { 'U' | 'u' } - sets the value of the mode switch to
UNINSTALL
debug ::= { 'D' | 'd' } - sets the value of the debug switch to ON
console ::= { 'C' | 'c' } - sets the value of the console switch to ON
service ::= { 'S' | 's' } [ '0' | '1' | ... ] – sets the value of the
service switch to ON, if '1' and to OFF if '0'. Otherwise the value will
remain the same.
verbose ::= { 'V' | 'v' } - sets the value of the verbose switch to ON
report-registry ::= { 'RR' | 'rr' } - sets the value of the
ReportRegistry switch to ON
report-license ::= { 'RL' | 'rl' } - sets the value of the ReportLicense
switch to ON
report-disable ::= { 'R-' | 'r-' } - sets the value of the ReportEnabled
switch to OFF
report-files ::= { 'RF' | 'rf' } - sets the value of the ReportFiles
switch to ON
report-datetime ::= { 'RT' | 'rt' } - sets the value of the
ReportDatetime switch to ON
name ::= { 'N' | 'n' } blank filename - defines the value of the name
variable
timeintervall ::= { 't1' | 'T1' } blank milliseconds - defines the value
of the timeinterval variable. Several consecutive time intervals can be
specified: -t1 3000 -t2 6000 -t3 9000 etc.
paramfile ::= { ‚E' | ‚e' } blank filename – defines the value of the
paramfile variable
filename ::= a string of characters without spaces
```

Multiple parameters can be specified at the same time, also repeatedly, and also those which cannot be used as a combination, e.g. -U ad /i. In such case the latter is used.

Use the following registry key to specify the parameter:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\'osecmdienst'\
```

The `ImagePath` string with the path to enaio® server as its value and the 'n' parameter with the service name is created during installation. The default entry follows the syntax:

```
C:\OSECM\server\axsvckrn.exe -n 'osecmdienst'
```

If you want to have the date and time information (ReportDateTime) added to the name of the report about the server start and license information be included in the report (ReportLicense) add the following value to the string:

```
C:\OSECM\server\axsvckrn.exe -n 'osecmdienst' -rt -rl
```

Main Thread

Until now, only the following happened: the user request has been understood and all switches have been set accordingly. Actual initialization can take place now. It is important to understand that in case of a service the SCM administers the MainThread. This is a delicate stage as the MainThread not being exited correctly will lead to the service neither being exited properly, and in case the server starts as application, it would not happen.

The process initialization included several successive steps. Each step has its own number (like an ID) and a name. Some steps just have technical meaning and others load licenses, determine the server ID, and implement the system logic. The number and order of steps can soon be changed.

In case any step was not successful, the initialization will be cancelled and the step number will be returned as an error code to the operating system and:

- a message box will open in case Verbose is set to ON
- a WindowsEventLog will be written
- a message will be sent to the administrator
- the startup will be logged
- a report will be written

All messages have a similar format and at least the cause known by the server is everywhere communicated. You can simply test that by changing the server ID in the table and restarting the server.

The administrator gets informed by e-mail or by Net message. This is done through the following registry entries:

- AdminMail indicates the administrator's e-mail address
- AdminNet is the name of the computer receiving the Net message

The e-mail is sent by MAPI.

In case initialization was successful and the server is running, a message is sent to the administrator, as well. The same will happen at server termination (after successful start).

Report

The report file contains all steps executed at server start. Therein, all errors and warnings, such as 'Unable to load executor', that do not stop the server launch but may be important can be found. 'Executor not loaded'. An unloaded executor would be considered as an error. In such a case, the kernel keeps running but the administrator must be informed as client programs will not run properly. Furthermore, there are warnings, e.g. 'this step has taken too much time'. Despite of that the kernel starts but communicates certain network problems or the like. Possibly, specific system files cannot be accessed. In fact, these are described in the table, but if the kernel checks the file's readability, the operating system will send negative answers. That may indicate file unavailability.

The e-mail to the administrator informs about the number of errors and warnings so it can be recognized instantly whether or not troubleshooting is required despite of successful server launch.

This number of errors is also contained in the report file.

This file is created at startup of the server on the server OSDEV

```
====== Protokoll vom Start des AppServers  ======
====== ausgef³hrt am 02/09/2002   ======
==========================================================
 701 - 14:59:46,723 | MainThread started
     | CommanLine : -n askrn
     | Instance : askrn
     | HasConsole : 0
     | IsService : 1
     | Verbose : 0
     | ReportRegistry : 0
     | ReportLicense : 0
```

```
    | ParamFile :
    | Temp Dir : D:\os\server\ostemp\
    | Root Dir : D:\os\server\
    | File Path : D:\os\server\axsvckrn.exe
    | Conf Path : D:\os\server\etc\
    | User Path : D:\os\server\etc\user\
    | ok: 0,0 sec
 703 - 14:59:46,723 | CoInitialize
    | ok: 0,0 sec
 705 - 14:59:46,723 | CreateMsgQueue
    | ok: 0,0 sec
 707 - 14:59:46,723 | ChangeAccess
    | ok: 0,0 sec
 709 - 14:59:46,723 | Activate PerfMonitor
    | ok: 0,0 sec
 711 - 14:59:46,723 | InitialLoadParams
    | CurrentSchema : 4.0
    | ok: 0,468 sec
 713 - 14:59:47,192 | Check OS Version
    | OS : NT 5.0 - Service Pack 2
    | ok: 0,0 sec
 715 - 14:59:47,192 | Init winsock
    | ok: 0,16 sec
 717 - 14:59:47,207 | Set Threads context parameters
    | ok: 0,0 sec
 721 - 14:59:47,207 | GetIdentStrings
---->    14:59:47,239  |  Win32API  Error.  Function:  <NetWkstaUserGetInfo>.
Parameter: <> Error: 0x520
    |  Eine  angegebene  Anmeldesitzung  ist  nicht  vorhanden.  Sie  wurde
gegebenenfalls bereits beendet.
---->    14:59:47,239  |  De  Server  startete  unter  LocalSystem  Account.
M÷glicherweise sind nicht alle Netzwerk Resources verf³gbar.
    | Address found : 195.127.121.49, 11D3-00B0D0F0C8CB
    | IP Address : 195.127.121.49
    | GUID : 11D3-00B0D0F0C8CB
    | TCP Port : 4010
    | Computer name : OSDEV
    | Domain name :
    | User name : SYSTEM
    | ok: 0,31 sec
 723 - 14:59:47,239 | DBConnect
    | DB Parameter:
    | DSN : peter400ms
    | USER : sysadm
    | Real DB Parameter:
    | DSN : peter400ms
    | UID : sysadm
    | APP : optimal_AS
    | WSID : OSDEV
    | DATABASE : peter40
    | LANGUAGE : Deutsch
    | ok: 0,31 sec
 725 - 14:59:47,270 | Server feststellen
    | SELECT id, group_id FROM server WHERE (name = 'OSDEV') AND (instance =
'askrn') AND (port = 4010)
    | ServerID : 5
    | ServerGroupID : 5
    | ok: 0,32 sec
 727 - 14:59:47,301 | check resources locked after crash
    |  DELETE  FROM  oslockedres  WHERE  (sessionguid  IN  (SELECT  ss.sessionguid
FROM ossession ss WHERE ss.serverid = 5))
    | DELETE FROM ossession WHERE serverid = 5
    | ok: 0,62 sec
 729 - 14:59:47,364 | Resource manager initialization: License
    | ok: 0,78 sec
 731 - 14:59:47,442 | Resource manager initialization: system files
---->  14:59:47,442 | Access to File c:\os\server\etc\as.cfg denied: 4
    | Das System kann den angegebenen Pfad nicht finden.
---->  14:59:47,442 | Access to File c:\os\server\etc\aslisten.dat denied: 4
    | Das System kann den angegebenen Pfad nicht finden.
    | ok: 0,16 sec
 733 - 14:59:47,473 | Servicename prüfen
    | ok: 0,0 sec
 735 - 14:59:47,473 | Expires prüfen
    | ok: 0,0 sec
 737 - 14:59:47,473 | Server - Lizenz prüfen
    | ok: 0,0 sec
 739 - 14:59:47,473 | Create ConnectThread
    | ok: 0,0 sec
 741 - 14:59:47,473 | Create ReadThreads
    | ok: 0,0 sec
 743 - 14:59:47,473 | Create ExLoadThread
    | ok: 0,0 sec
 745 - 14:59:47,473 | Create SD
    | ok: 0,0 sec
 747 - 14:59:47,473 | Create PeriodicBatches
    | osrevisit : abn.CheckOsrevisit, 3000
    | WFM : wfm.WorkerJob, 5000
    | ok: 0,0 sec
 749 - 14:59:47,473 | Create PeriodicThread
    | ok: 0,15 sec
 751 - 14:59:47,489 | Create JobQueues
    | common : 4, -1
    | wfm : 2, -1
    | deferred : 1, 100
    | ok: 0,0 sec
 753 - 14:59:47,489 | Create NameSpaces
    | abn : common, oxjobabn.dll
    | adm : common,
    | dbp : common, oxjobdbp.dll
    | krn : common,
    | kts : common,
    | lic : common,
    | std : common, oxjobstd.dll
    | tst : common, oxjobtst.dll
    | wfm : wfm, oxjobwfm.dll
    | ok: 0,16 sec
 755 - 14:59:47,504 | Initialize Executors
    | abn : ok
    | adm : ok
    | dbp : ok
    | krn : ok
    | kts : ok
    | lic : ok
    | std : ok
    | tst : ok
    | wfm : ok
    | ok: 0,219 sec
 757 - 14:59:47,723 | Initialize Performance Collector
    | ok: 0,31 sec
 763 - 14:59:47,754 | Create JobThread: common, 0
    | ok: 1,281 sec
 763 - 14:59:49,036 | Create JobThread: common, 1
    | ok: 1,266 sec
 763 - 14:59:50,301 | Create JobThread: common, 2
    | ok: 1,250 sec
 763 - 14:59:51,551 | Create JobThread: common, 3
    | ok: 1,250 sec
 763 - 14:59:52,801 | Create JobThread: wfm, 0
    | ok: 1,250 sec
 763 - 14:59:54,051 | Create JobThread: wfm, 1
    | ok: 1,250 sec
 759 - 14:59:55,302 | Create SendThread: 0
    | ok: 0,0 sec
 761 - 14:59:55,302 | Resume Threads
    | ok: 0,0 sec
==========================================================
====== Protokoll abgeschlossen
====== Fehler: 0, Warnungen: 4
```

Hence, the number, time, and duration of the step are logged, errors and warnings are noted.

The Server Console

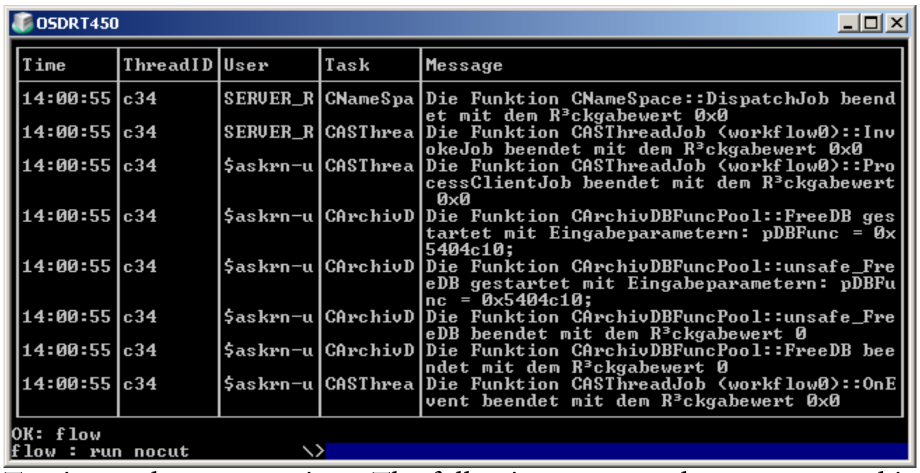
If you start the server with the parameter 'console', it will create a console in which all actions executed by the server and error messages are displayed. You can use the console to enter/send commands to the server.

The console window is divided into several parts. The first part of the bottom window line displays the operating mode of the console. After a prompt, you can enter commands into this line. The second last window line displays text and status (Ok or error) of the most recently executed command.

The top part of the console window displays all information on the respective operating mode.

You can run the console in one of the following modes:

- Flow

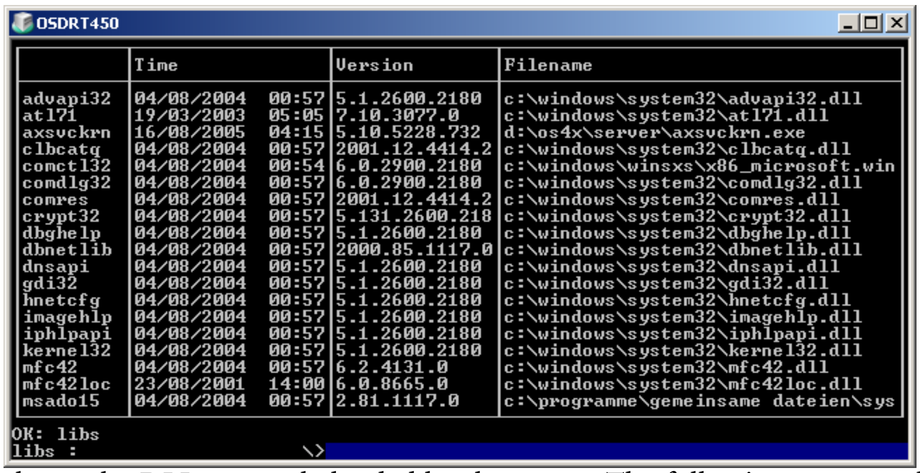


```
OSDRT450
Time     |ThreadID|User     |Task     |Message
14:00:55 |c34     |SERVER_R |CNameSpa |Die Funktion CNameSpace::DispatchJob beend
         |        |         |         |et mit dem R³ckgabewert 0x0
14:00:55 |c34     |SERVER_R |CASThrea |Die Funktion CASThreadJob (workflow0)::Inv
         |        |         |         |okeJob beendet mit dem R³ckgabewert 0x0
14:00:55 |c34     |$askrn-u |CASThrea |Die Funktion CASThreadJob (workflow0)::Pro
         |        |         |         |cessClientJob beendet mit dem R³ckgabewert
         |        |         |         | 0x0
14:00:55 |c34     |$askrn-u |CArchivD |Die Funktion CArchivDBFuncPool::FreeDB ges
         |        |         |         |tartet mit Eingabeparametern: pDBFunc = 0x
         |        |         |         |5404c10;
14:00:55 |c34     |$askrn-u |CArchivD |Die Funktion CArchivDBFuncPool::unsafe_Fre
         |        |         |         |eDB gestartet mit Eingabeparametern: pDBFu
         |        |         |         |nc = 0x5404c10;
14:00:55 |c34     |$askrn-u |CArchivD |Die Funktion CArchivDBFuncPool::unsafe_Fre
         |        |         |         |eDB beendet mit dem R³ckgabewert 0
14:00:55 |c34     |$askrn-u |CArchivD |Die Funktion CArchivDBFuncPool::FreeDB bee
         |        |         |         |ndet mit dem R³ckgabewert 0
14:00:55 |c34     |$askrn-u |CASThrea |Die Funktion CASThreadJob (workflow0)::OnE
         |        |         |         |vent beendet mit dem R³ckgabewert 0x0
OK: flow
flow : run nocut           \>
```

Tracing and error warnings. The following commands are supported in flow mode:

pause or set pause 1	turn off message display
run or set pause 0	turn on message display
cut or set cut 1	- cut message
no cut or set cut 0	– continue message in next line
clear	- delete display in console

- Libs

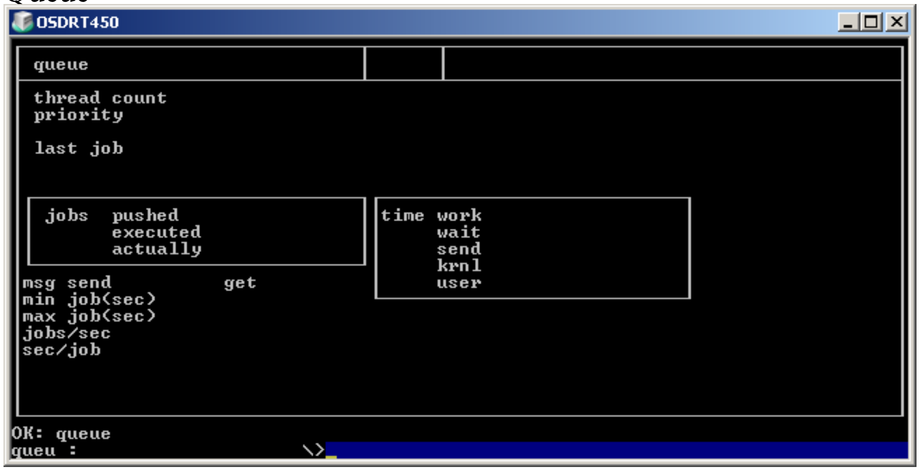


```
OSDRT450
         |Time               |Version        |Filename
advapi32 |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\advapi32.dll
atl71    |19/03/2003   05:05 |7.10.3077.0    |c:\windows\system32\atl71.dll
axsvckrn |16/08/2005   04:15 |5.10.5228.732  |d:\os4x\server\axsvckrn.exe
clbcatq  |04/08/2004   00:57 |2001.12.4414.2 |c:\windows\system32\clbcatq.dll
comctl32 |04/08/2004   00:54 |6.0.2900.2180  |c:\windows\winsxs\x86_microsoft.win
comdlg32 |04/08/2004   00:57 |6.0.2900.2180  |c:\windows\system32\comdlg32.dll
comres   |04/08/2004   00:57 |2001.12.4414.2 |c:\windows\system32\comres.dll
crypt32  |04/08/2004   00:57 |5.131.2600.218 |c:\windows\system32\crypt32.dll
dbghelp  |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\dbghelp.dll
dbnetlib |04/08/2004   00:57 |2000.85.1117.0 |c:\windows\system32\dbnetlib.dll
dnsapi   |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\dnsapi.dll
gdi32    |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\gdi32.dll
hnetcfg  |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\hnetcfg.dll
imagehlp |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\imagehlp.dll
iphlpapi |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\iphlpapi.dll
kernel32 |04/08/2004   00:57 |5.1.2600.2180  |c:\windows\system32\kernel32.dll
mfc42    |04/08/2004   00:57 |6.2.4131.0     |c:\windows\system32\mfc42.dll
mfc42loc |23/08/2001   14:00 |6.0.8665.0     |c:\windows\system32\mfc42loc.dll
msado15  |04/08/2004   00:57 |2.81.1117.0    |c:\programme\gemeinsame dateien\sys
OK: libs
libs :                     \>
```

shows the DLL currently loaded by the server. The following commands are supported in the libs mode:

refresh or update	-refresh DLL list

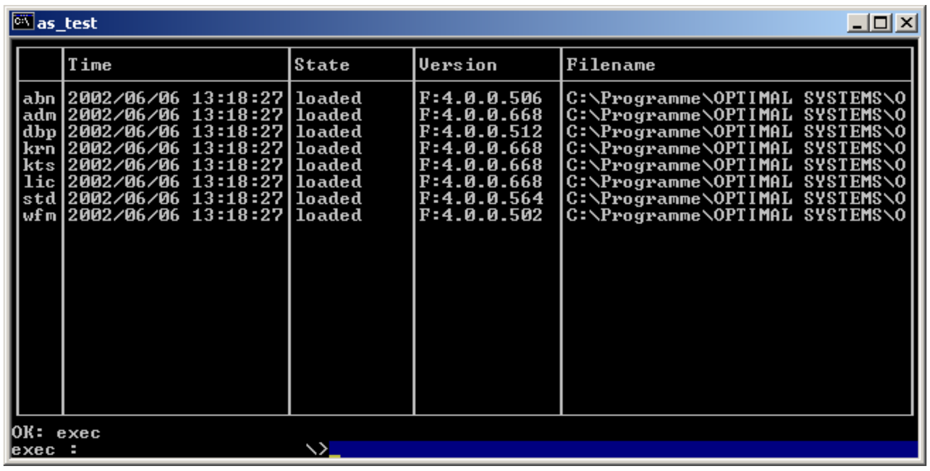
- Queue



```
OSDRT450
 queue

 thread count
 priority

 last job

  jobs  pushed              time  work
        executed                  wait
        actually                  send
msg send            get           krnl
min job(sec)                      user
max job(sec)
jobs/sec
sec/job

OK: queue
queu :                     \>_
```

displays current state of job queues;

- Exec

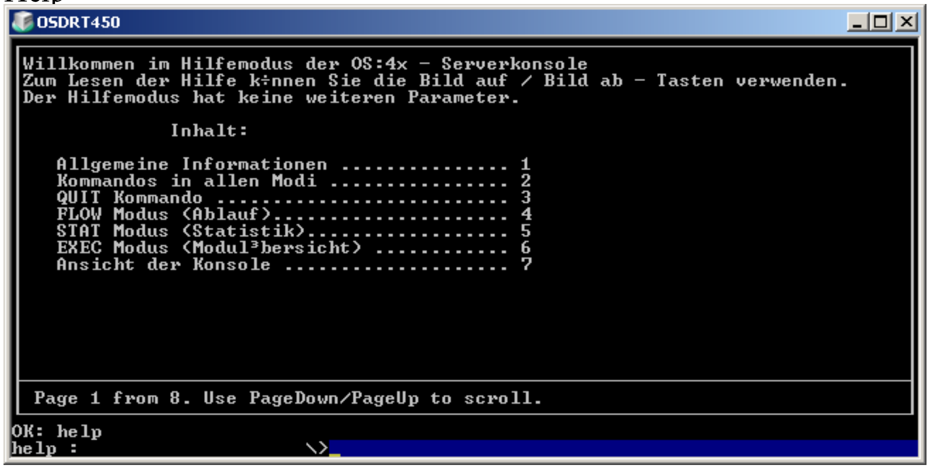


```
as_test
    Time                State   Version     Filename
abn 2002/06/06 13:18:27 loaded  F:4.0.0.506 C:\Programme\OPTIMAL SYSTEMS\O
adm 2002/06/06 13:18:27 loaded  F:4.0.0.668 C:\Programme\OPTIMAL SYSTEMS\O
dbp 2002/06/06 13:18:27 loaded  F:4.0.0.512 C:\Programme\OPTIMAL SYSTEMS\O
krn 2002/06/06 13:18:27 loaded  F:4.0.0.668 C:\Programme\OPTIMAL SYSTEMS\O
kts 2002/06/06 13:18:27 loaded  F:4.0.0.668 C:\Programme\OPTIMAL SYSTEMS\O
lic 2002/06/06 13:18:27 loaded  F:4.0.0.668 C:\Programme\OPTIMAL SYSTEMS\O
std 2002/06/06 13:18:27 loaded  F:4.0.0.564 C:\Programme\OPTIMAL SYSTEMS\O
wfm 2002/06/06 13:18:27 loaded  F:4.0.0.502 C:\Programme\OPTIMAL SYSTEMS\O

OK: exec
exec :                     \>_
```

shows list of loaded executors. The following commands are supported in the exec mode:
load <executor> -load <executor>
unload <executor> -remove <executor>
updates the list of loaded executores after each command;

- Stat
  not implemented

- Help



```
OSDRT450

Willkommen im Hilfemodus der OS:4x - Serverkonsole
Zum Lesen der Hilfe k÷nnen Sie die Bild auf / Bild ab - Tasten verwenden.
Der Hilfemodus hat keine weiteren Parameter.

             Inhalt:

   Allgemeine Informationen ............... 1
   Kommandos in allen Modi ................ 2
   QUIT Kommando .......................... 3
   FLOW Modus (Ablauf)..................... 4
   STAT Modus (Statistik).................. 5
   EXEC Modus (Modul³bersicht) ............ 6
   Ansicht der Konsole .................... 7

 Page 1 from 8. Use PageDown/PageUp to scroll.
OK: help
help :                         \>_
```

help for implemented operation modes

Enter the mode name into the command line to call the required operation mode. For help mode you can enter help, h or ?.

In all modes, the commands q, quit or exit are allowed. Their entry exits the console and closes the window. Having started the server as usual application, it will exit together with the console exit. Otherwise, the service continues running.

In the INI file axsvckrn.ini you can set up the console configuration (color scheme, width, display of information fields, etc.). If the file has been changed while the console was running, you can apply the changes to the current display by entering cfg (in all operation modes).

Example of axsvckrn.ini:

```
[Console]
CodePage=1251
ActiveMode=flow
```

Check List in case of Problems at Server Start

The following pages will inform you on how to proceed in case enaio® server cannot start.

Server

The application server disposes of different mechanisms in order to perform a detailed error diagnose.

The following options exist:

- Start with start.bat as an application
- Error display with verbose mode
- Logging of the start and shutdown sequence

Start with start.bat as an application

A batch file which starts the server as an application is located in the server directory. The server is then executed under the account of the logged-on user. The logging steps are displayed in the console during startup.

If the server starts as an application and not as a service permission problems are likely. Verify the account used to start the service.

Error display with verbose mode

If you start the server with the command line parameter -v, error messages are displayed at initialization in a message box. With the help of the error list you can find out what to do in specific situations.

Logging during server start

Independently from the logging over oxrpt.dll, the initialization steps can be written to a separate log file that is saved into the server directory. The file's default name is startup_YYYYMMDD_hhmmss.txt and logs the single steps. Therein, you can find possible error causes.

Check list of possible actions

Symptom	Possible cause	Actions
Service does not start	Miscellaneous	Read log in OS temp directory Interpret error message according to table below Execute start.bat
License error	License file does not exist or invalid	Export and check current license file with axlicimpexp. - station recognition, service name, IP address
DB error	Configuration error in the database, possibly due to database modification or transfer to other computer	By means of the registry entries Comstring and IPPort (CurrentSchema) + service name a valid entry in the server table is searched. If necessary, adjust entries in the database or registry.
DB-Connect fails	DSN not set up	Check, whether there is a system DSN. The name is found in the DB section of the registry.
	Wrong user name/password	Change password with Axchgpwd.exe
	Missing network permissions	Change the account under which the service is running
Jobs fail partially	Executors are not loaded	Use the console to check whether executors are loaded or not. Alternatively you can analyze the start up log.

Error codes during server start

Returncode	Int	Initialization step	Description	Possible cause/activity
SRETCODE_START	701	MainThread started	Starts main thread, initiate internal variables	→ DEV
SRETCODE_COINITIALIZE	703	CoInitialize	Windows- COM library is initialized	Check service pack, check permissions of the server account, otherwise → DEV
SRETCODE_CREATEMSGQUEUE	705	CreateMsgQueue	Internal initialization of the message queue	→ DEV
SRETCODE_CHANGEACCES	707	ChangeAccess	Initialization of the Windows Process	Check account for service (grant locale

S			Security	admin rights)
SRETCODE_ACTIVATEPERFMON	709	Activate PerfMonitor	Activating the performance monitor	→ DEV
SRETCODE_LOADPARAMS	711	InitialLoadParams	Registry parameters are loaded	Check registry Keys: schemata, current schema (value), current schema (key) must exist. Check read and write rights of the registry
SRETCODE_CHECKOS	713	Check OS Version	Operating system check	→ DEV
SRETCODE_WINSOCK	715	Init winsock	Initialization of the network level	Check service pack, TCP IP setting
SRETCODE_THREADCONTEXT	717	Set Threads context parameters	Internal initialization	→ DEV
SRETCODE_CREATECONSOLE	719	Create Console	Console creation if specified in the start parameters	Check settings in service manager 'Allow interactive data exchange with desktop'
SRETCODE_IDENTSTRINGS	721	GetIdentStrings	Determine computer name, IP address, GUID, user name, domain name	Check SP Check network configuration, check account
SRETCODE_DBCONNECT	723	DBConnect	Database is connected	Check DSN, registry (DB parameter), connectivity, account
SRETCODE_SERVERID	725	Determine server	Server ID and group OD is read from the database	Check consistency of table server and server group. If necessary, registry (COM string, port, and service name are relevant), see SQL statement in report
SRETCODE_CHECKRES	727	check resources locked after crash	Locked resources are released after the last server crash	Check database connection, check OSLockedRes in the structure, see SQL statement in report
SRETCODE_RESMGRLIC	729	Resource manager initialization: License	Reading and encryption of the license	License possibly wrong imported, check table oslicense, export and check license file with axlicimpexp if necessary
SRETCODE_RESMGRFILES	731	Resource manager initialization: system files	System files of OS resources are determined and their existence is checked	Check the table osressources, check path specifications for resources with type = 1 if necessary
SRETCODE_CHECKNAME	733	Check Servicename	Service name from the license is compared with real service name	License and service name Service manager (properties)
SRETCODE_CHECKEXPIRES	735	Check Expires	Expiration of the license file is checked	Check license file and system time
SRETCODE_CHECKLICENSE	737	Check server license	Server license is verified	Check assignment of the station to the server in aslic.cfg/in axliccfg.exe or in the table oslicressources, change station number if required

SRETCODE_CREATECONNECT SRETCODE_CREATECONNECT	739	Create ConnectThread	Initialization of the connect thread for reception of client connections	TCP setting and check whether port is occupied (e.g. through already running server instances)
SRETCODE_CREATEREAD	741	Create ReadThreads	Read threads are created	→DEV
SRETCODE_CREATEEXLOAD	743	Create ExLoadThread	Executor loading thread is created	→DEV
SRETCODE_CREATESD	745	Create SD	Security Descriptor is created	Check account, then → DEV
SRETCODE_CREATEPEBATCHES	747	Create PeriodicBatches	Periodic jobs are initialized	Check registry [Currentschema]\Batches → DEV
SRETCODE_CREATEPERIODIC	749	Create PeriodicThread	Thread for timer is created	→ DEV
SRETCODE_JOBQUEUES	751	Create JobQueues	JobQueues are created depending on the registry settings	Check registry [CurrentSchema]\Queues
SRETCODE_NAMESPACES	753	Create NameSpaces	Namespaces are read from the registry and initialized	Check registry [CurrentSchema]\NameSpaces Assign queue, DllName correct
SRETCODE_INITEXEC	755	Initialize Executors	Executors are initialized	Check if DLLs are available, DLL version must match Kernel
SRETCODE_INITPERFCOLL	757	Initialize Performance Collector	Internal initialization of the system environment	→ DEV
SRETCODE_CREATESEND	759	Create Send thread	Threads for notifications are initialized	→ DEV
SRETCODE_RESUME	761	Resume threads	Threads are resumed (have not been started previously)	→ DEV
SRETCODE_CREATEJOB	763	Create Job threads	Job threads for editing are created	Check registry Check registry [CurrentSchema]\Queues (NumberOfThreads and Priority) → DEV

Please note that the number of steps are also the return values which are displayed by the control manager if a service could not be started.

Returncode	Int	Description	Cause
SRETCODE_REPORTINIT	790	The logging system could not be activated	Oxrpt.dll not available, not configured
SRETCODE_SERVICEINDEBUG	791	The service cannot be started in debugging mode	Replace Debug - exe with release
SRETCODE_WRONGSYSTEM	792	The archive server cannot be executed on this operating system.	Check the operating system.
SRETCODE_SYSTEMFAILED	793	The execution of a standard function returned an error	Report file to DEV
SRETCODE_ALREADYRUNNING	794	A server with the same service name is already running	Check task manager, check registry
SRETCODE_INVALIDPARAMETER	795	The command line parameters are invalid	Check command line
SRETCODE_TLSSLOT	796	TLS slot could not be determined	-> DEV
SRETCODE_TAKEINSTANCENAME	797	The service name could not be locked.	-> DEV

SRETCODE_GETBREAKEVENT	798	The break event could not be created	-> DEV

Adminmail and Net-send

The enaio® server can now send a notification to enaio® administrator when starting or stopping the service. This can be done by e-mail or Net-send. If the notification is sent by e-mail, the files startup.txt and shutdown.txt will be attached. The e-mail can either be sent via MAPI (requires the installation of a MAPI client) or via SMTP.

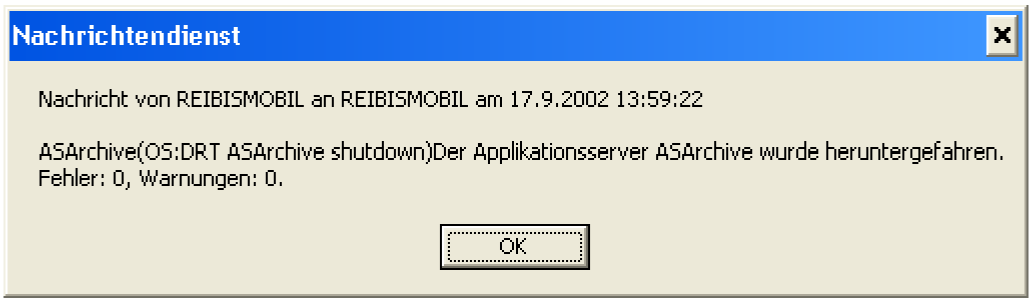
To configure the server respectively, the following registry entries must be set. By default no values are set, i.e. if no entry exists, no notification takes place.

Under the key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Optimal Systems\ASArchive\Schemata\4.0
```

String	Value	Description
AdminMail	e.g. reibis@optimal-systems.de	E-mail address of the recipient
AdminNet	e.g. reibismobile	Computer name of the recipient for net send
MailSender	e.g. robert	The sender of the e-mail must be specified when sending an e-mail
MailServer	e.g. osapp	Name of the mail server. If this name is not specified, it is attempted to send via MAPI, else via SMTP.

Net-send message at shutdown:

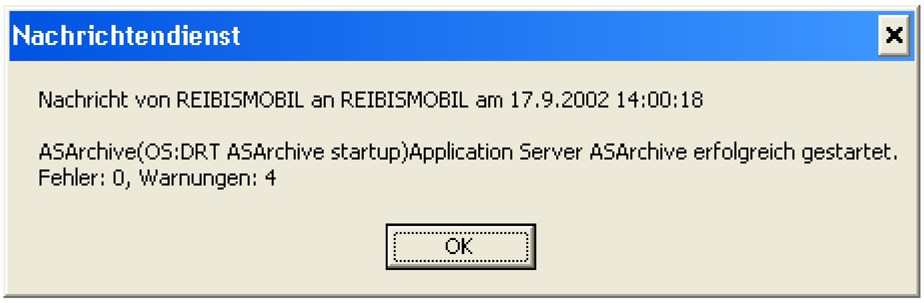

Nachrichtendienst

Nachricht von REIBISMOBIL an REIBISMOBIL am 17.9.2002 13:59:22

ASArchive(OS:DRT ASArchive shutdown)Der Applikationsserver ASArchive wurde heruntergefahren.
Fehler: 0, Warnungen: 0.

Figure 62

Net-send message at startup:


Nachrichtendienst

Nachricht von REIBISMOBIL an REIBISMOBIL am 17.9.2002 14:00:18

ASArchive(OS:DRT ASArchive startup)Application Server ASArchive erfolgreich gestartet.
Fehler: 0, Warnungen: 4

Figure 63

How to Select the Interface Language in enaio®

The following list combines how to select the interface language of the enaio® components. Components that are not listed here do not allow switching the interface language.

enaio® component	Language selection
enaio® client `ax.exe`	The language of enaio® client is set in the settings dialog that you can open using the SETTINGS function on the ENAIO ribbon tab. Select the language from the 'SELECT LANGUAGE...' workspace. The language of the object definition used (provided that it has been translated) is selected in the same dialog under OBJECT DEFINITION LANGUAGE (Sprache der Objektdefinition). Note that some areas of the interface will be translated after having restarted the program, while, since they depend on the Windows operating system, others will be shown in the language which was set for the operating system. enaio® client is also available as Web application (enaio® webclient). The language of enaio® webclient is selected in the MS Internet Explorer under TOOLS › INTERNET OPTIONS › GENERAL › LANGUAGES. If the language of the used operating system is neither German nor English, the interface of enaio® webclient will be shown in English.
enaio® administrator `axadmin.exe`	The program uses the language settings of enaio® client.
enaio® enterprise-manager `osecm_entmgr.msc`	The language is specified in the file `oxentmgr.cfg` which is located in the `..\clients\admin` directory. To change the language, remove the semicolon from the LOCALIZATION entry of the current language and prepend it to the LOCALIZATION entry of the language to be used.
enaio® editor `axgredit.exe`	The program uses the language settings of enaio® client.
enaio® capture `asindex.exe`	The language is set on the ENAIO ribbon tab using the WORKSPACE function.
enaio® editor_for Workflow `axwfedit.exe`	The program uses the language settings of enaio® client.
enaio® workflow-administrator `axwfadm.exe`	The program uses the language settings of enaio® client.
enaio® office-utilities `oxvbofficeutil.dll`	The language of OS_office-utilities is selected in the MS Office programs. The LANGUAGE SETTINGS icon available on the enaio® ribbon allows changing the interface language.
enaio® document-storage `axvbdocstorage.exe`	The language is selected by clicking on the LANGUAGE icon in the context menu of the icon in the taskbar.
enaio® manager-for-logfiles `axprotocolcfg.exe`	The language for log configuration is selected under ? › SELECT LANGUAGE (Sprache wählen). Afterwards the program must be restarted.

enaio® protocol-viewer `axrptview.exe`	The language is selected in the log file viewer under VIEW › SELECT LANGUAGE... (Ansicht > Sprache wählen...).
enaio® file-trigger `axfiletrigger.exe`	The program uses the language settings of the Windows operating system. If the language of the used operating system is neither German nor English, the interface of enaio® file-trigger will be shown in English.
enaio® file-system-archiver `axfsarch.exe`	The program uses the language settings of enaio® client.
enaio® contentviewer	The language is selected via the language settings in the MS Internet Explorer under TOOLS › INTERNET OPTIONS › GENERAL › LANGUAGES.
enaio® portable `axvbportable.exe`	The language of enaio® portable is selected under FILE › LANGUAGE.
enaio® portable-client `axvbreader.exe`	In enaio® portable-client, the language is selected under VIEW › SETTINGS › WORKSPACE 'SELECT LANGUAGE...'.
enaio® server	The language can be changed to English in the following Registry entry: HKEY_LOCALE_MACHINE/SOFTWARE/ OPTIMAL SYSTEMS/'SERVICE'/locale=1033