otris software AG

EMBEDDED ARCHIVE STORAGE (EAS)
System Documentation

DOCU VERSION 1.0.9
JUNE 27, 2014

otris software AG
Königswall 21
D-44137 Dortmund

www.otris.de

Table of Contents

1. Introduction

Embedded Archive Storage (EAS) is an archive software that can be integrated into any applications (below referred to as host systems). This software generally does not include its own graphic user interface (GUI) and is executable on client computers and as server software on the network.

Below we explain the EAS structure and the most important settings options. It is assumed that EAS is integrated into the host system's setup. This guide is a tutorial for system integrators, showing what a sample setup looks like.

This guide uses an ongoing example of an archive named `store1`, which is used to describe step by step all settings and tasks for setting up EAS.

> ***Important note***
>
> *Before you set up an EAS system using this documentation, please read this document **carefully first.** Particularly during initialization, some steps might, in case of doubt, not be easily repeatable. Moreover, there are some requirements for the operating system prior to installation which must be fully met prior to installing EAS.*

2. System Components

The functions for archiving and search/retrieval are accessed by the host system via a Web service. This Web service is a REST interface that is transported via http protocol. For a description of the REST interface, please refer to the Integration guide.

Fig. 1 shows the system structure, including the system components Web service(EAS_HTTPD), the location for management data (EAS_RW), and the actual archive repository (EAS_WORM).

In EAS the archive repository location is explicitly separate from the management data location because both storage media are faced with very different requirements.

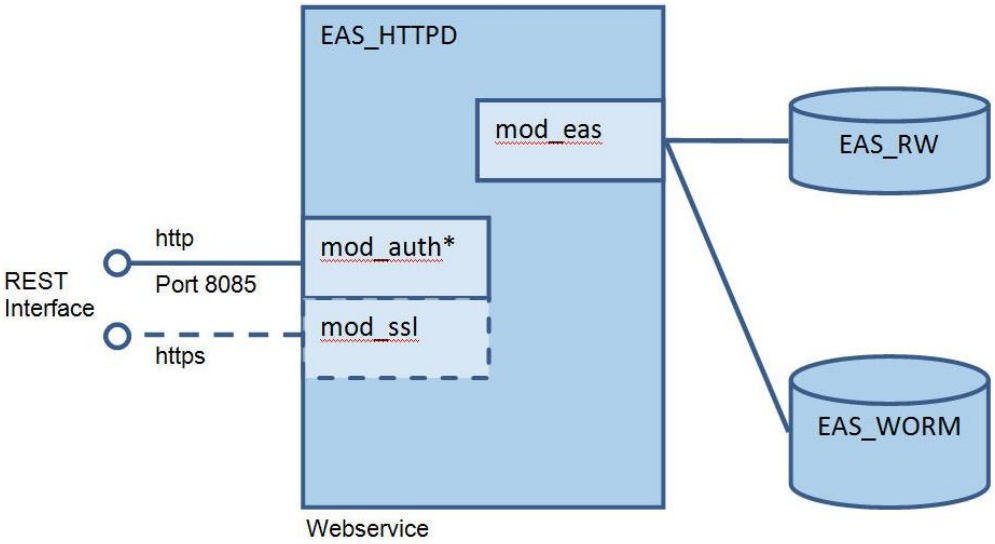

Fig. 1: EAS system structure

2.1 Web service (EAS_HTTPD)

The EAS Web service is the key element for providing the archive and search/retrieval functionality. It is implemented as a Windows service (EAS_HTTPD). The requests and responses are made available to the connected host system using a communications port (preset to 8085) via http protocol.

2.2 Location for management data (EAS_RW)

The EAS management data is composed of the following components:

- Index: Directory for full text search of archived data files, as well as feature and range search on index fields of DOCUMENTS files
- Registry: Table of contents where all archived data files of the archive are registered
- Spooler: Cache for data files to be archived
- Audit Trail: Logs all accesses to the archive
- Policies: Storage location for access and retention policies

The management data is required for ongoing archive operation. This involves numerous accesses here because the individual parts are eventually read or continued to be written for each user action on the archive.

Therefore, the primary requirement for the storage media used for the management data is to guarantee quick read and change access to all data anytime.

2.3 Location for archive repository (EAS_WORM)

The archive repository is a high-security area for revision-proof archived data. This location is arranged in such a manner that data can be created and queried by reading there, but are not changed (**W**rite **O**nce **R**ead **M**any).

Unlike the EAS_RW area of the management data, it is primarily important in the WORM repository that enough space is available in a storage array designed for data security.

3. System Requirements

The EAS archiving functionalities are always embedded as a component into a host system. The result is that the hardware requirements are, of course, closely tied to those of the host system.

Generally, the EAS component can run on 32-bit and 64-bit Windows systems from Windows XP Professional.

3.1 Software

3.1.1 Operating system

It is recommended that you run the software on modern Windows Server operating systems. In the case of DOCUMENTS as a host system, EAS can be used on all Windows operating systems where DOCUMENTS is available.

3.1.2 Installed components

To set up the service and to run the command line tool, you need a runtime environment for Visual C++. It must be installed prior to EAS, and is available from Microsoft as a "Redistributable package" (vcredist.exe). This package can be installed under Windows and you do not have to subsequently load other components via a network connection.

Optionally, the Windows utility iFilter can be used for the EAS full text filter. Its use in combination with EAS requires the runtime environment of the .Net Framework version 3.5. This environment is also available from Microsoft; it is automatically registered in Windows.

3.2 Hardware

On one hand, the hardware requirements affect the computer's architecture regarding processor, RAM, etc. while, on the other hand, they affect a storage media design.

3.2.1 Processor and server architecture

Overall, you need to be aware that the recommendations for computer design are highly dependent on the host system requirements and its operating structure. In multi-user operation of an ERP system, for instance, CPU and RAM must be larger in size than in a smaller desktop version.

We recommend a modern multi-processor computer with at least 4 GB RAM, where performance increases by using more modern and larger sized hardware.

3.2.2 Storage media

EAS stores data in different areas for management data (EAS_RW) and archive data (EAS_WORM). In the context of the system component, this has already been described (see chapter 2).

The hardware requirements are highly dependent on the size and number of documents to be archived and on the computer load (e.g. many concurrent requests with many users).

Therefore, we can only make basic recommendations here which in individual cases must be subject to closer analysis:

- EAS_RW: A RAID 1 array from a size of 25 GB is recommended.
  You should be able to access as quickly as possible the management data, search index and registry stored here.
- EAS_WORM: A RAID 5 array or a storage system (NAS, Netapp, FAST LTA Qube, etc.), which can be integrated into the file system, is recommended.

4. Installation

The host application in which EAS is embedded is responsible for providing a setup (msi file and the like).

In an example, we explain an installation here where the "`eas`" directory is the root directory of the installation. The `eas` root directory can, however, also be copied to any other location, as required from the viewpoint of the host application. Note that the directories below this root directory are neither renamed nor is their hierarchy changed. Fig. 2 shows installation of the `eas` root directory in the default folder `Programs`.

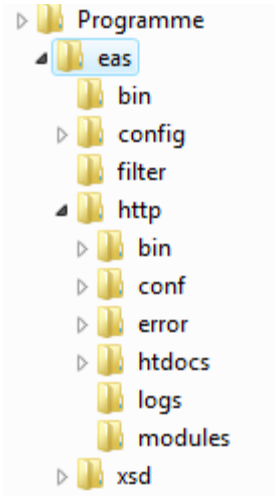

Fig. 2: EAS installation directory hierarchy

The chapters below explain the contents of the included subdirectories.

4.1 [INSTDIR]\bin

The binary code file required for running the Web service and for running the command line tool are stored here.

The Web service is implemented as Apache module. The `mod-eas.dll` file contains the module's implementation, which is integrated into the Web service.

The eas.exe file is a command line tool that enables, among others, creating and initializing new archives. You also need to execute maintenance tasks via this command line tool. The "`eas.exe --help`" call provides an overview of the deployed functions.

When using the command line tool, ensure that the executing user has administrator rights (this can be set in the UAC on Windows) and that the name of the working directory is `[INSTDIR]\bin`.

Moreover, this directory contains dynamic runtime libraries (dll) which are required both by the Web service and the `eas.exe` command line tool.

These DLLs come with EAS only; the Visual C++ 2008 Redistributable package (`vcredist.exe`), to be installed beforehand, does not already contain them.

4.2 [INSTDIR]\Config

The specific configurations of the archive(s) are stored in this subdirectory.

The `eas.conf` file is started from the Web server's base configuration; it is used to integrate one (typical) or more archives. In the preinstalled sample configuration, an archive named "`store1`" is integrated. Other archives can be integrated accordingly.

Configuration of the "`store1`" archive, which is referred to in Web service configuration by the "EASSetConfigFile" directive, is performed in the configuration file "`store1.ini`".

4.3 [INSTDIR]\http

Installation directory of the included Web server Apache http, version 2.2.

This is the original directory structure of the Apache distribution. The executables are identical to the original files, only the configuration (`[INSTDIR]\http\conf\httpd.conf`) has been customized. Only the modules required to run the EAS module are integrated and shipped with the software (see `[INSTDIR]\http\modules`). The content of this directory can be replaced with an Apache Web server preconfigured on the system, if necessary. It is important that the EAS-specific `[INSTDIR]\config\eas.conf` file be integrated into the `http.conf` file.

4.4 [INSTDIR]\filter

This directory contains the full text filters that come with the software.

These filters are applications to which the data file to be searched is transferred as a parameter and which return the included full text (e.g. relevant content parts of a PDF file) to the standard version. Should the iFilterReader.exe file that comes with the software be used, the .Net Framework 3.5 must first be installed as runtime environment on the target system.

The iFilterReader does not itself extract the text on the files, but for this purpose it reverts to the iFilters of the Windows search that exist on the respective computer. To be able to index popular Microsoft Office documents, the "Microsoft Office 2010 Filter Packs" [1] can be used. They contain filters for earlier Office versions (97-2003; .doc, .ppt, .xls) as well as for Metro Office (2007; .docx,

[1] http://www.microsoft.com/download/en/details.aspx?id=17062

.pptx, .xlsx). Moreover, they contain filters for Zip, OneNote, Visio, Publisher and for the Open Document formats.

The Filter Packs are available as 32-bit and 64-bit downloads and are available, respectively, for the languages English, German, French, Spanish, Italian, Russian, Japanese, and Chinese.

The following message types are supported: Windows 7, Windows Server 2003 Service Pack 2, Windows Server 2008 R2, Windows Server 2008 Service Pack 2, Windows Vista Service Pack 1, Windows XP Service Pack 2, and Windows XP Service Pack 3.

To determine which filters are already installed on a computer, you can use the free software tool iFilter Explorer[2].

4.5 [INSTDIR]\xsd

This folder contains the XML schema definitions (XSD) for the different XML representations which come with the software:

- XML representation of a DOCUMENTS file to be archived: `externRecord.xsd`
- XML representation of an archived DOCUMENTS file: `internRecord.xsd`
- XML representation of the search hit list: `rss.xsd`
- All other XML representations: `eas.xsd`

The remaining XSD files are embedded by those already mentioned.

These XSD files can be used for validating the XML representations while developing the host application.

[2] http://www.citeknet.com/Products/IFilters/IFilterExplorer/tabid/62/Default.aspx

5. Initializing and Configuring the Archive

The archive configuration serves both the Web service as information source for the data to be archived and to initialize the archive as initially required.

5.1 Archive settings

Default configuration of an archive is extremely simple. The code area below displays the default configuration for the `store1` archive, and then explains the settings of the individual areas. The configuration in this format is already complete and can be initialized. Changes – particularly to the storage structure within the file system – should be made in any event. At least the drive names and directories in the `[variables]` section should be customized to the RAID arrays or storage systems used. The configuration values to be changed for this are highlighted in red.

```
#************************************************************
#
#                              store1.ini
#
#          Typical archive configuration for store1
#
#
#************************************************************

[variables]
;Unique archive name (without spaces and special characters)
ARCHIVE_NAME=store1

;Directory for quick read and write access (RW)
;for search index, registry and spooler, which in emergencies
; can be restored from the high security area
RW_DIR=C:/EAS_RW

;Directory/drive for the high security area (WORM)
;where the revision-proof data is stored.
;A backup concept must exist for this area!
WORM_DIR=E:/EAS_WORM

[archive]
archiveRoot=${WORM_DIR}/${ARCHIVE_NAME}/ARCHIVE
policyPath=${RW_DIR}/${ARCHIVE_NAME}/POLICIES
spoolRoot=${RW_DIR}/${ARCHIVE_NAME}/SPOOL
; Optional: Path to the license file, in case it does not reside next to
the ini file
; or its name is not license.pem
; license=${WORM_DIR}/license.pem

[audittrail]
path=${RW_DIR}/${ARCHIVE_NAME}/AUDIT

[index]
path=${RW_DIR}/${ARCHIVE_NAME}/INDEX

[registry]
path=${RW_DIR}/${ARCHIVE_NAME}/REGISTRY/registry.db

[signature]
keypath=${WORM_DIR}/${ARCHIVE_NAME}/KEYS

[filter]
pdf=utf-8;../filter/pdftotext.exe -q -enc UTF-8 %d -
```

```
txt=cmd.exe /c "type %d"
;pdf=utf-8;../filter/IFilterReader.exe /file:%d /ext:PDF
;tif=utf-8;../filter/IFilterReader.exe /file:%d /ext:tif
;tiff=utf-8;../filter/IFilterReader.exe /file:%d /ext:tif
;DOC=utf-8;../filter/IFilterReader.exe  /file:%d /ext:DOC
;DOCX=utf-8;../filter/IFilterReader.exe  /file:%d /ext:DOCX
;XLS=utf-8;../filter/IFilterReader.exe  /file:%d /ext:XLS
;XLSX=utf-8;../filter/IFilterReader.exe  /file:%d /ext:XLSX
;PPT=utf-8;../filter/IFilterReader.exe  /file:%d /file:PPT
;PPTX=utf-8;../filter/IFilterReader.exe  /file:%d /ext:PPTX
```

The configuration shown here shows, in the `[variables]` section, the minimum settings that need to be made for an EAS archive:

- Each archive must have a unique name. That name may not contain any umlauts or special characters; it must be used in a URL (ARCHIVE_NAME).
- You need to specify a directory for the archive's management data (index, table of contents, spooler, access log and policies) (EAS_RW). In this path, a subdirectory with the archive name is automatically created, so the value can be specified unchanged for multiple archives if this part of the file system has enough storage capacity.
- You need to specify a directory for the high security area (the repository) where the data to be archived is stored (EAS_WORM). In this path a subdirectory with the archive name is also automatically created, so the value can be specified unchanged for multiple archives if this part of the file system has enough storage capacity.
- It is recommended that you specify a path to a valid license file. If the license file information is missing, a file named license.pem will be searched for in the directory of the respective ini file. EAS cannot be launched if this file is not present or the license is invalid.

For the `store1` configuration that the standard package contains, the following directories would be created through initialization:

```
c:\EAS_WORM\store1\ARCHIVE
c:\EAS_WORM\store1\KEYS
c:\EAS_RW\store1\INDEX
c:\EAS_RW\store1\AUDIT
c:\EAS_RW\store1\SPOOL
c:\EAS_RW\store1\POLICIES
c:\EAS_RW\store1\REGISTRY
```

Below we will discuss advanced settings that impact on the behavior of the system components defined in the sections.

> ***Important note on initializing the archive***
>
> *What is important is that these settings be made* ***prior t****o* initializing the archive and that they remain unchanged while executing the Web service.
>
> *Once made, settings such as encrypting the archived file*s cannot be easily modified during the lifetime of an archive.

5.1.1 Section [variables]

Here you can define variables that are valid in the further course of the data file and that are referenced with `${Variablenname}`. Nested referencing in the `[variables]` section is *not* allowed.

The most important system parameters are set as an example in this section in the above shown store1.ini file. These are the name of the archive (ARCHIVE_NAME) and the names of the two directories for data storage (EAS_RW and EAS_WORM). All other settings can be made in the remaining sections using these parameters.

5.1.2 Section [archive]

The settings for the archiving operation are made here.

In the `store1.ini` sample, the following directories are entered:

- `archiveRoot`: Points to a directory in the file system which is the high security area (the repository) of the archive. All data files to be archived are stored in this directory. This area of the file system can be implemented and used as WORM medium.

- `policyPath`: Indicates where the archive's access and retention policies are retained. Because these policies may change, they are stored, by default, in the management data section of the archive.

- `spoolRoot`: Points to a directory in the management data section where attachments can be temporarily saved during the archive operation.

- `license`: Points to a valid license file. If this information of the license file is missing, the system will be looking for a file named license.pem in the Apache Server's working directory. EAS cannot be launched if this file is not present or the license is invalid.

- `annotationsPath`: Points to a directory in which the annotations to DOCUMENTS files are stored. If a path is not specified, the annotations will be stored in `archiveRoot` in the *annotations* subdirectory.

- `flagsPath`: Points to a directory in which the flags for the DOCUMENTS files are stored. If no path is specified, the flags will be stored in the same folder as the DOCUMENTS files.

- `additionalRecordFields`: This optional switch sets whether information on the initial archiver and the initial archiving time are also inserted for new file versions. This switch is initialized with `yes`.

Settings for versioning

With regard to versioning archived data files, EAS provides two different variations that are defined via the optional `versioningMode` setting:

- Simple versioning (`simple`): New archived file versions are saved completely, even if no changes have been made to individual attachments. This variation is the default; it is used without explicitly specifying the `versioningMode` setting.
- Differential versioning `(differential)` **:** Only the XML file of the archived file and the *changed* attachments are saved in the new version. The new XML file refers to unchanged attachments.

The following setting allows changing the type of versioning from simple to differential.

```
versioningMode=differential
```

Only the newest archived file versions are found respectively in the default behavior of the **search**. If the earlier versions should also be found in the search, this can be set through the following property:

```
searchOnlyNewestVersion=no
```

Optional setting for encryption

The `encryptArchive` setting is used to control that the archive data is stored encrypted in the high security area (repository).

This is not encrypted in the default setting. When selecting this option, you must additionally define the type of encryption in the section `[encryption]` described below.

Example of enabling encryption:

```
encryptArchive=yes
```

Optional setting for the Spooler

The `spoolCleanup` setting is used to define that an attachment be removed from the Spooler directory by the EAS service after the archive operation. The default setting is `yes`, and the attachment is thus removed. Whereas if deletion is to be performed using the host application itself, you will need to select `no` for this setting.

Example of turning off cleaning the Spooler through EAS:

```
spoolCleanup=no
```

Another option is to store the temporarily stored attachments in the Spooler directory encrypted as well. This parameter must be considered in combination with `encryptArchive`; it inevitably presumes the `[encryption]` section.

In default behavior, the attachments are *not* stored encrypted in the Spooler. The following example illustrates the required setting to store the attachments temporarily stored in the Spooler encrypted:

```
encryptSpool=yes
```

Other settings

EDA has an internal mechanism to control retention periods for DOCUMENTS files; however, it is not used by DOCUMENTS. It can be enabled during search to improve performance. The following switch is available for this purpose:

```
forceNeverMaxExpire=yes
```

Optional settings for the storage strategy

The data files can be stored in the repository according to different storage strategies. Examples of this are chronological storage according to archive time, or storage according to management information from the archived file.

The `storageMode` parameter allows setting the storage strategy in the repository. The following modes can be set:

- `archiveDateTime` (default): Archived files are stored chronologically in directories based on the archive date. For instance, a DOCUMENTS file archived at 10:20 AM on February 20, 2010 is stored in the `2010/02/20/10/<fileid>` subdirectory.
- `fieldBased`: The storage directory is dynamically determined from the archived file's index fields. This allows using, in the `storagePattern` parameter, general management information (_id, _masterId, _type, _creator, _creatorLogin, _lastModifier, _lastModifierLogin, _archiver, _archiverLogin, _version, _createDateTime, _lastModifiedDateTime, _archiveDateTime, _title), and also specific file fields via the `%a` reference.
- `hashBased`: The save operation is uniform. The optional parameters `storageDepth` (default: 2) and `storageNameLength` (default: 2) determine the depth of the subfolder structure or the length of these directory names.

Example of logical storage (`fieldBased`):

```
storageMode=fieldBased
storagePattern=%a_type%/%_aid%
```

Example of a uniform storage strategy (`hashBased`):

```
storageMode=hashBased
storageDepth=2
storageNameLength=2
```

Additionally, parts of timestamp values can also be specified as names of folders with a field value-based storage strategy. For this purpose, the `$year()`, `$month()`, `$day()`, `$hour()`, `$minute()` and `$second()` functions are available in the `storagePattern` switch. The functions return the corresponding part of the timestamp.

Example of the folder structure `<Year>/<File ID>`:

```
storagePattern=$year(%a_archiveDateTime%)/%a_id%
```

If the passed value is not a timestamp, the value will be returned unchanged by these functions.

Other optional settings

For all responses of the REST interface returning an XML file a stylesheet can be subsequently embedded by specifying the `stylesheet` parameter. This can, for instance, be used for lightweight client solutions.

Example:

```
stylesheet=/eas.xsl
```

You can set that the archive attribute is set under Windows for all archived files in the file system. To do this, you have to turn on the `archiveBitAttribute` switch.

```
archiveBitAttribute=on
```

5.1.3 Section [audittrail]

The user accesses to the archive are logged in the Audit Trail.

The respective settings for the directory path of the logs and their scrolling can be set in the associated section of the initialization file:

The `path` parameter points to a directory in which the log file(s) are stored. By default, rolling log files are used that write a new log file every day.

The `rollingType` setting lets you influence at which intervals a new log file is begun. Allowed values are `Hourly`, `Daily` (default), `Weekly`, `Monthly`, `Yearly`.

Example of weekly creation of a new log file:

```
rollingType=Weekly
```

5.1.4 Section [index]

The settings for indexing the archived files including all attachments are made here.

The `path` parameter points to the directory in the EAS management data where the search index is saved. This directory should enable quick read and change file accesses.

Optional settings to the indexing operation:

`indexRecord` allows setting that the fields of the archived files should be imported into the search index.

The useful default setting is `yes`. Whereas `no` allows preventing transfer to the search index, i.e. you can only read archived files directly from the archive, based on their ID.

The `indexAttachment` setting lets you set that the content of an attachment should be imported into the search index. Again, the useful default setting is `yes`. Whereas `no` allows preventing transfer to the search index. This means that

archived files are no longer retrievable on the basis of the content of their attachments via full text.

> ***Important***
> *The prerequisite for correct import into the search index is to define a suitable filter in the* `[filter]` section.

The `delayedRecordIndexing` parameter optionally enables subsequent indexing.

In default behavior `delayedRecordIndexing` is set to `no` and the fields of the archived files are automatically transferred to the search index during the archive operation. This indexing can be subsequently performed time-controlled for time-sensitive archive scenarios using the command line tool `eas.exe` (see chapter 7: "Maintenance"). The required setting for the parameter then is:

```
delayedRecordIndexing=yes
```

Analogous to the fields, the `delayedAttachmentIndexing` parameter allows subsequently indexing the attachments.

When setting this parameter to `yes`, the contents of the attachments will not be automatically transferred to the search index as full text during the archive operation. This indexing can be subsequently performed time-controlled for time-sensitive archive scenarios using the command line tool `eas.exe` (see chapter 7: "Maintenance"). The default is `no`, i.e. the full texts are transferred to the search index already during the archive operation.

> ***Note***
> *If subsequent indexing of DOCUMENTS files has been configured, this will also affect attachments.*

Optional settings for full text indexing of attachments:

The `stopwordlist` parameter allows limiting the individual words from the full texts of the attachments using a connected text file. Trivial words defined in this list, such as "the", "a" or "an", are *not* imported into the search index, thereby significantly easing its load. By default, a stopword list is *not* used.

Example of using a stopword list:

```
stopwordlist=C:/Program Files/EAS/config/stopwords_utf8_de.txt
```

The optional parameter `minimalWordLength` additionally determines from which length (number of letters in a word) full text terms are imported into the index. The default is `0`, i.e. each word is imported. The following example incorporates all words that consist of at least three letters in the index:

```
minimalWordLength=3
```

If a stopword list is additionally defined, only words with a minimum length of three letters *and not* on this list will be imported into the index.

The `maximal`WordLength parameter determines up to which length full text terms are imported into the index. The default is `65536`, i.e. every word is, as it were, imported.

Example of limiting indexing to words with a maximum length of 255 characters:

```
maximalWordLength=255
```

This parameter sets in which characters a string to be indexed is segmented into individual words. The initial value is

```
nonTokenChars= \r\n\t\v\f;,
```

i.e. strings are separated in all line breaks, tabulators, page breaks and vertical line breaks, as well as semicolons and commas.

nonChars sets which characters are removed from a word in the search index. "(abc"), for instance, would be converted to "abc", so the search queries "(abc)" and "abc" would return the same result. The default is:

```
nonChars=".,%#*+^:\"\';()-
```

Optional settings for the type of indexing

The `type` switch allows setting the manner in which indexing takes place. The following variations are available:

- `CLuceneStandard`: Indexing type, which opens the Indexer for each index entry, and then closes it again.
- `CLuceneRecordBatch`: Indexing type, which collects all index entries for a DOCUMENTS file first, and then writes these bundled in the index.

The default is `CLuceneStandard`.

`CLuceneRecordBatch` can be faster in specific scenarios because all write processes for a DOCUMENTS file are written bundled in the index. Whereas `CLuceneStandard` is a bit safer because the index is only opened for a single write operation, and then immediately closed. This reduces the likelihood that the index can enter an inconsistent state, e.g. through computer downtime.

Optional settings for CLucene

The `clucene_maxSegmentsMergeSize` switch sets the maximum size, in MiB, at which a segment can be merged. If a segment is larger, it will no longer be merged with other segments. If no value is defined, segments of up to any size will be merged.

```
clucene_maxSegmentsMergeSize=2
```

The `clucene_minSegementsMergeSize` switch sets the minimum size, in MiB, at which a segment is always merged if it is smaller than that size. Default value: 0.

```
clucene_minSegmentsMergeSize=1
```

The `clucene_maxOptimizeSegments` switch sets the maximum number of segments to remain after an optimization process. Default value: 1.

```
clucene_maxOptimizeSegments=1
```

The `compoundFormat` switch sets whether the parts of a segment are always to be saved in a summarized file (`yes`) or not (`no`). Default value: `yes`. Saving in individual files may slightly increase performance during indexing because summarizing is not applicable.

```
compoundFormat=yes
```

The `clucene_maxClauseCount` switch sets into how many internal subqueries a query can be resolved. The limit is to prevent that, particularly in the case of large search indices, too much RAM is used by a search query. When this limit is exceeded, EDA generates an error message "TooManyClauses". The value 1024 is preset.

```
clucene_maxClauseCount=1024
```

Optional settings for indexing versions

The `indexingMode` switch sets how versions of a DOCUMENTS file are captured. The following modes are available:

- `allVersion`: All versions of a DOCUMENTS file are collected in the index.
- `newestVersion`: Only the newest version of a DOCUMENTS file is collected in the index.

The default is the `allVersions` mode.

```
indexingMode=newestVersion
```

> ***Note!***
> *When you use the indexing mode `newestVerions`, the `searchOnlyNewestVersion` switch in the `[archive]` section is without effect.*

Other optional settings

Delimiters for indexing can be customized via the `nonTokenChars` switch in the [index] section. Spaces (Carriage Return, Newline, Vertical TAB and TAB) are always separators. However, you can define others using `nonTokenChars` . This switch is initialized with ";,", i.e. texts are also segmented into tokens with semicolons and commas. Whether the comma is a decimal separator of a number is taken into consideration here, though. In that case there will be no segmentation into tokens.

```
nonTokenChars=;,.
```

5.1.5 Section [registry]

The settings for the archive's table of contents (registry) are made here.

The type switch allows setting the registry's type. The following two types are allowed:

- `Sqlite3`: The registry is saved in an Sqlite3 database.
- `Sqlserver`: The registry is saved in an MS SQL Server database.

`Sqlite3` is preset.

Optional specific switches for Sqlite

The `path` parameter points to the directory in the EAS management data where the registry is saved. This directory should enable quick read and change file accesses.

```
path=/EAS/store1/registry.db
```

The `sqlite3_pagesize` switch set the page size, in bytes, that Sqlite 3 uses for saving. Default value: 1024.

```
sqlite3_pagesize=1024
```

For explanations, see SQL documentation at
http://www.sqlite.org/pragma.html#pragma_page_size

The `sqlite3_cachesize` switch sets the size, in pages, of the internal cache of Sqlite. Default value: 2000.

```
sqlite3_cachesize=2000
```

For explanations, see SQL documentation at
http://www.sqlite.org/pragma.html#pragma_cache_size

The `sqlite3_journalmode` switch sets in which mode the transaction logs are written. The following values are allowed:

- `truncate`: The save operation is performed in a separate file. After completion the file is reset to size 0.
- `memory`: The transaction logs are saved in RAM.
- `wal`: Uses write-ahead log instead of rollback transaction logs. This is a specific write mode that is faster and more "parallel" than the rollback transaction logs, given the same security.

Default value: WAL.

```
sqlite3_journalmode=truncate
```

> *For explanations, see SQL documentation at http://www.sqlite.org/pragma.html#pragma_journal_mode*

Specific switches for MS SQL Server

The `sqlserver_name` switch sets the name for the ODBC connection to the SQL server.

```
sqlserver_name=localhost
```

An ODBC connection of the same name must be set up via ODBC configuration.

> ***Important note***
>
> *There are problems with some versions of the ODBC implementation version "SQL Server" (e.g. 6.02.9200.16384). An error message saying "Invalid precision value" appeared here! This was due to errors in ODBC implementation. These problems could not be detected with ODBC implementation "SQL Server Native Client", which is why its use is recommended.*

The `sqlserver_user` switch sets the user name for the ODBC connection.

```
sqlserver_user=username
```

The `sqlserver_password` switch sets the password for the database connection.

```
sqlserver_password=password
```

5.1.6 Section [signature]

Settings that define the type of signing the data to be archived are made here.

`signaturePath` points to a directory that contains the keys for signing the archived files. These keys are generated on initializing the archive. You really have to retain them and protect them from being accessed by third parties. They are therefore, by default, stored in the high security area of the archive (EAS_WORM).

Optional settings

The `hashAlgorithm` allows defining the algorithm to be used for creating the signature. Allowed values are `sha1`, `sha256`, `sha384`, `sha512` and `md5`. The default is `sha256`, which today is considered sufficiently secure.

5.1.7 Section [encryption]

In this optional section those settings are made for storing the data files to be archived encrypted in the repository. It is encrypted synchronously with a modified AES algorithm via the EAS service.

The `keyMode` parameter is used to set that the encryption is controlled via a `password` or a `file` using a key. In both cases the `keySource` parameter contains information about the key. The command line enables creating a new key. This must really be safely retained and protected from third-party access:

```
cmd> [INSTDIR]\eas.exe --generate-key 256 -c [INSTDIR]\estore2.ini
```

Example of AES encryption via a password

```
keyMode=password
keySource=mysecretpassword
```

Example of AES encryption via a file key

```
keyMode=file
keySource=C:\Program Files\eas\config\encryption\key
```

5.1.8 Section [filter]

All applications that allow filtering the contents from attachments are deployed here.

For instance, the `pdftotext.exe` application allows extracting texts stored in PDF files. The command line tool `type` in turn extracts text from simple text files.

These filters can be defined for specific file extensions, i.e. the file type of the attachments. The file to be filtered, "%d", is the filter input. Filters are, according to convention, output to the default output.

```
[filter]
pdf=utf-8;../filter/pdftotext.exe -q -enc UTF-8 %d -
txt=cmd.exe /c "type %d"
```

Using the iFilter on Windows

The so-called iFilters are suited to extract the contents of attachments on Windows systems. They are used by the operating system to index the data files for search in Explorer. The program shipped with the software uses these iFilters and routes the information obtained to the search index.

When using this mechanism, you need to ensure that the .Net Framework Runtime 3.5 has already been initially installed on the server and that the corresponding iFilters are available and registered:

```
[filter]
DOC=utf-8;../filter/IFilterReader.exe  /file:%d /ext:DOC
DOCX=utf-8;../filter/IFilterReader.exe  /file:%d /ext:DOCX
XLS=utf-8;../filter/IFilterReader.exe  /file:%d /ext:XLS
XLSX=utf-8;../filter/IFilterReader.exe  /file:%d /ext:XLSX
PPT=utf-8;../filter/IFilterReader.exe  /file:%d /file:PPT
PPTX=utf-8;../filter/IFilterReader.exe  /file:%d /ext:PPTX
tif=utf-8;../filter/IFilterReader.exe /file:%d /ext:tif
tiff=utf-8;../filter/IFilterReader.exe /file:%d /ext:tif
```

5.1.9 Section [performanceMonitor]

Monitor type

type can be used to set how to save the measurements. The following types are available:

- `Null`: The measurements are not saved. This is equivalent to a disabled performance monitor.
- `File`: The measurements are saved in a file. You can use `outputPath` to specify the location. The files are written in here. Each hour is set to the rolling file appender . The names of the files follow the YYYY-MM-DD-HH.csv schema.

The Null type is set here as the default.

Output path

The `outputPath` switch specifies the folder in which the files are written using the measurements.

> ***Important!***
> *Nothing is set preset, so the files are written in the EAS server's working directory. However, it is recommended that you explicitly specify a folder here.*

Example of a configuration where the measurements are written in the `/PerfLogs` folder:

```
[performanceMonitor]
Type=File
outputPath=/PerfLogs
```

Threshold

You can set from how many measurements these are written in the files. You set this via the `Threshold` switch. The value 10000 is preset here, i.e. only after 10000 measurements have been entered will these 10000 be written in the file. Until then, these are held in memory.

This switch allows better checking of the influence of writing the measurements in the files, because this also has an influence on the performance of the overall system.

A sample configuration, where all 50000 measurements are written in the log file, looks as follows:

```
[performanceMonitor]
Type=File
outputPath=/PerfLogs
threshhold=50000
```

Log file

The written files are CSV files that consist of five columns:

1. Process ID
2. ID of DOCUMENTS file or of spool document
3. Time of measurement in ISO-8601 format
4. Security Token of the performing user
5. Code for the executed action (see below)
6. Time, in milliseconds, elapsed since starting the respective process
7. Comments and additions to the process, e.g. reference to spool processes during archiving

The action codes in the fifth column have the meaning as listed in the following table.

Action code	Meaning
0	Start archive operation
1	Indexing an archive file/creation complete
2	Converting the IsNewestVersion flag complete
3	Reindexing an existing archive file complete
4	Deleting the Spool files complete
5	Storage for a archive file/creation in the WORM domain complete
6	Entry in registry for a archive file/creation
7	File signature complete
8	Preparing the archiving operation complete
9	Archive operation complete
10	Starting DOCUMENTS file update
11	DOCUMENTS file update complete
12	Starting spooling
13	Spooling complete
14	Starting search
15	Search complete
16	Starting setting lock flag
17	Setting lock flag complete
18	Starting querying DOCUMENTS file/attachment
19	Querying DOCUMENTS file/attachment complete
20	Starting setting deletion flag
21	Setting deletion flag complete
22	Starting deleting lock flag
23	Deleting lock flag complete
24	Starting deleting deletion flag
25	Deleting deletion flag complete
26	Opening the IndexWriter
27	Closing the IndexWriter
28	Determining storage location
29	Written data in registry
30	Assemble index data for document
31	Starting creating index document

32	Adding fields to index document
33	Considering access policy
34	Adding index document to index
35	Starting setting annotation
36	Setting annotation complete
37	Starting reading annotation
38	Reading annotation complete
39	Starting deleting annotation
40	Deleting annotation complete
41	Starting version query (previous)
42	Version query (previous) complete
43	Starting version query (next)
44	Version query (next) complete
45	Starting creating a directory
46	Creating a directory complete
47	Requesting Mutex for writing index

5.1.10 Section [excludedIndexFields]

Fields not to be indexed are entered in this section. Internal file fields are not considered here. In the sample configuration, the file field "field1" is excluded from the index.

```
[excludedIndexFields]
field1=yes
```

5.2 Initialization

Before starting a Web service, you need to create the archive / the archives using the following command:

```
[INSTDIR]\bin\eas.exe –i –c [INSTDIR]\config\store1.ini
```

It is necessary to ensure that the command call is started using administrator rights and write permissions on the directories used (User Account Control). The configuration file `store1.ini` contains all settings required to run the archive. Essentially, this is defining the storage directories and properties, such as storage encryption or storage.

6. Installing and Configuring the Web Service

As explained in the previous chapters, the Apache HTTP Server, version 2.2 is used as the platform for the REST Web service. For this reason, it is also included in a reduced version; however, it can be replaced with a preinstalled version on the system. Parallel operation of two and more Web servers on a system is also possible. Here you need to ensure that the respective ports are used for the http/https protocol. Below we will refer to the version that ships with the software. The default http port here is 8085.

6.1 Basic Web service configuration

The basic Web service configuration is performed in the `[INSTDIR]\http\conf\http.conf` file. This configuration is synchronized to running the Web server using the EAS module; it only includes the most necessary settings.

Setting the http port is important because there may be no overlapping with other systems on the computer. The default 8085 may need to be changed via the list directive.

```
# Lists: Allows you to bind Apache to specific IP addresses and/or
# port, instead of the default. Default for EAS is Port 8085
Lists 8085
```

Besides the absolutely necessary modules `auth_basic`, `authn_file`, `authz_host`, `authz_user` and `mime_module`, three more modules can be optionally integrated for diagnostics purposes:

```
# Optional Server Request Logging in logs/access.log
#LoadModule log_config_module modules/mod_log_config.so

# Optional Server Diagnostics
LoadModule status_module modules/mod_status.so
LoadModule info_module modules/mod_info.so
```

Configuration of these modules is carried out below in conditional directives.

If the REST service should be SSL encrypted via https, the ssl module must be integrated and configured accordingly. For details, see documentation on the Apache http Service.

```
# Optional Usage of https instead of http
#LoadModule ssl_module modules/mod_ssl.so
…
#
# Secure (SSL/TLS) http connections (only if mod_ssl is loaded)
#
# Use oppenssl to generate self-signed certificates:
#    openssl genrsa -out key.pem 2048
#    openssl req -x509 -days 365 -new -out cert.pem -key key.pem
#            -config http/conf/openssl.cnf
#
<IfModule ssl_module>
   SSLRandomSeed startup builtin
   SSLRandomSeed connect builtin
   SSLEngine on
   SSLCertificateFile    conf/ssl/cert.pem
   SSLCertificateKeyFile conf/ssl/key.pem
</IfModule>
```

At the end of the `http.conf` file the `[INSTDIR]/config/eas.conf` file including the EAS-specific additions to the http basic configuration is included, and explained in the chapter below.

```
# ************************************************************************
# Include Settings for Embedded Archive Server (EAS)
# ************************************************************************
Include ../config/eas.conf
```

6.2 Integrating the archive

To run the REST Web service via the Apache http Server, you need to integrate two modules that are not part of the Apache distribution:

1. The `apreq2` module is used to split multipart requests.
2. The `eas` establishes the connection to the EAS archive core.

The `[INSTDIR]\config\eas.conf` file, included by the Apache http basic configuration, loads these modules and includes all archive configurations from the `[INSTDIR]\config\stores\` directory.

```
#************************************************************
#
#                            eas.conf
#
# Web service configuration of Embedded Archive Server (EAS)
#        (included from [INSTDIR]\http\conf\http.conf)
# ************************************************************

# Loading the APREQ2 module to split the multi-part requests
LoadModule apreq_module ../bin/mod_apreq2.so
SetInputFilter apreq2

# Loading the module to connect EAS
LoadModule eas_module ../bin/mod_eas.dll

# Allowed, maximum size of a request; in this case, 2 GiB. Change
# both parameters to allow or enforce larger or
# smaller requests.
LimitRequestBody 2147483648
APREQ2_ReadLimit 2147483648
```

```
# Integrate the archives
Include ../config/stores/*.conf
```

6.3 Configuration files of stores

The structure of an individual configuration file for integrating a store into the Apache http Server is as follows:

```
<Location /eas/archives/store1>
      AuthType Basic
      AuthName "EAS_AUTH"
      AuthBasicProvider file
      AuthUserFile ../config/users.txt
      Require valid-user
      SetHandler eas
      EASSetConfigFile ../config/stores/store1.ini
</Location>
```

The archive defined and initialized via `store1.ini` (see chapter 5.1.10) is published under `/archive/store1`. It is important that you use the eas module to resolve this *location* and that you make a reference to the archive configuration of store1. The `EASSetConfigFile` switch is used to set the path to the ini file belonging to the store. Here you need to ensure that, for a relative path, this is relative to the Apache directory. Other archives can be integrated accordingly by using other location directives (e.g. under `/eas/archives/store2`).

If access to the archive should be protected, you can achieve this using the mechanism of the Apache Web server's basic authentication. In this case, the user information can be found in the

`[INSTDIR]\config\users.txt` file:

```
eas_user:$apr1$2ML5isH4$ZkrE0wyDoZ/lLMKYm9Om81
eas_administrator:$apr1$R2OzlLix$e0Ln068gLbVkNDK7Qo2N01
eas_keeper:$apr1$WuVmAWlq$/XNa1XATIj0933GhWHePJ0
eas_guest:$apr1$p8.BHkzU$YUCfeSEqKEvsKJN.CqJs5.
```

Login to the host system is always via so-called "Trusted users" (see White paper, chapter "Access Control"). For this purpose, the four roles `eas_user`, `eas_guest`, `eas_keeper` and `eas_admin` were defined. A user is created for each of these four roles. This is performed using the `htpasswd.exe` application from the Apache distribution (under `[INSTDIR]\http\bin`). The default setting supplies the `users.text` file including these users and the respective password "*system*". You should change these when creating the archive. The example below illustrates this for the user `eas_user`:

```
[C:\Program Files\eas\http\bin] htpasswd.exe -b ..\..\config\users.txt
eas_user mypassword
```

You need to proceed accordingly with the users `eas_guest`, `eas_keeper` and `eas_administrator`.

6.4 Installation

After performing the http service's basic configuration and integrating the archives, you need to initially register the Web service as a Windows service. To do this, you need to be an administrator, and the command line interpreter must be launched accordingly as an administrator. The Web service is registered as a Windows service under the name "`eas_http`" using the following command. It is important that you use the Apache root directory http as the working directory.

```
[C:] cd C:\Program Files\eas\http
[C:\Program Files\eas\http] .\bin\httpd.exe –k install –n "eas_http"
```

If registration as a Windows service has been successful, you can start this service now via the corresponding dialog in the Windows system settings or via the command line (run as an administrator) using:

```
[C:] net start eas_http
```

and stop it using the following command:

```
[C:] net stop eas_http
```

The Web service can now run and the `store1` archive can be reached under the following URL:

`http://myhostname:8085/archives/store1`

7. Maintenance

7.1 General maintenance

This system is actually maintenance-free. The administrator only needs to ensure that there is enough space on the storage media both for the management data and for the repository containing the archive data. In case of the management data, you need to keep an eye on the access protocol (audit trail) in particular because this may become very long, or many files are created with a rolling file appender setting.

7.2 Web service error log

Should there be problems or Web service failures, possible errors will be logged in the

```
[INSTDIR]\http\logs\error.log
```

file. If you want a different path and varying degrees of detail for the `error.log` file, this can be set in the basic configuration of the Apache http server (`[INSTALLDIR]\http\conf\http.conf`):

```
#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel error

<IfModule log_config_module>
    # The following directives define some format nicknames for use with
    # a CustomLog directive (see below).
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      # You need to enable mod_logio.c to use %I and %O
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\" %I %O" combinedio
    </IfModule>

    # The location and format of the access logfile
(common/combined/combinedio).
    CustomLog "logs/access.log" combined
</IfModule>
```

More error messages that might appear when starting the Web service can also be viewed in the Windows Event Viewer.

7.3 Consistency check of an archive

Archives can be subject to consistency checks. These determine whether archived data files were changed or deleted. If errors are detected, these will be displayed via the default output.

```
[C:\Program Files\eas\bin] eas.exe --verify-report -c ..\config\store1.ini
```

Starting the consistency check can be performed by the Windows Scheduler, for example, to enable it to automatically start at a specific time.

Optional switch to control the consistency check

When restoring an index, the time during which the DOCUMENTS files are to be restored can be limited. The beginning is specified using the call parameter `--time-period-begin`, the end using `--time-period-end`.

The time must be specified according to ISO 8601, e.g. like

```
--time-period-begin 2012-08-28T9:30
```

7.4 Subsequent indexing

As described in chapter 5.1.4, archived files and attachments can also be archived without initially incorporating them into the search index. This behavior is recommended for scenarios where many transactions must be transferred to the archive in a very short time.

The settings in the archive configuration (ini file) are:

```
delayedFileIndexing=yes
```

and/or

```
delayedAttachmentIndexing=yes
```

To update the search index at a later time, the following command line call can be used:

```
[C:\Program Files\eas\bin] eas.exe --update-index -c ..\config\store1.ini
```

Using this, all data files registered in the archive table of contents (registry) that have not yet been indexed are subsequently indexed. Automatic navigation of this command can also be performed by the Windows Scheduler. It is necessary to ensure that this command be executed via a command line or in Scheduler via administrator privileges.

7.5 Index tuning

The index consists of so-called segments. The number of segments and the size of segments varies over time because new segments are constantly created and existing ones are merged. This is due to technical factors, and enables writing in the index and reading from the index at the same time.

However, merging segments may take some time with large stores, e.g. sizes of several MiB to GiB, which may cause considerable delays with individual archiving

operations. To avoid this, `clucene_maxSegmentsMergeSize` can be used to limit the size. Merging to larger segments, which in turn facilitates faster searching, can be performed via the following command line call:

```
[C:\Program Files\eas\bin] eas.exe --optimize-index -c ..\config\store1.ini
```

> ***Important!***
> *Even if performing index tuning while running the EAS server is generally possible, it is recommended that you shut down the EAS server for tuning the index. The index is locked for tuning runtime, so all other write operations have to wait until the operations trigger timeouts. The same is true of updating and rebuilding the index.*

Optional switches for index tuning

The `--max-segments <NUMBER>` switch specifies the maximum number of segments to remain after a merge. Default value: `1`.

The `--max-merge-segments-size <NUMBER>` switch specifies the maximum size at which a segment is merged. The default value is the corresponding setting from the store's configuration file. Overwrites the setting from the configuration file.

7.6 Index status

The index status of DOCUMENTS files and attachments in the registry can be explicitly set or reset via a command line call:

```
[C:\Program Files\eas\bin] eas.exe --set-index-status 1 -c
..\config\store1.ini
```

Thus, for instance, re-indexing selected DOCUMENTS files or attachments can be forced. For the "Indexed" status, you have to specify the value 1, for the "Not-indexed" status, the value 0.

Moreover, this switch can be combined with the `--no-attachments`/`--no-record` and `--time-period-begin`/`--time-period-end` switches. See also section 9.3.

7.7 Registry migration

The registry can be migrated from the Sqlite3 database to a database in the MS SQL Server. To do this, you have to customize the store's ini file by entering the necessary data for accessing the database in the MS SQL Server there (see chapter 5.1.5).

Migration can then be started using the following command line call:

```
[C:\Program Files\eas\bin] eas.exe --migrate-registry /path/to/registry.db
-c ..\config\store1.ini
```

The database in the MS SQL Server must exist. If it contains a table labeled `files`, migration will then fail with a corresponding error message. In this case, you have to delete the table manually or use another database. The table will then be created and populated during migration.

The relevant switch is `--registry`. For a value, you need to specify the path to the previous Sqlite3 database.

8. Backup

Backup breaks down into four sections that are described in the chapters below:

1. You need to back up archive and Web service configuration directly after installation. These also include the private and public keys for signing and the key for optional encryption of archive data.
2. The management data is required for production mode of the archive component. Backup during operation is difficult because this data is subject to constant change.
3. The data to be archived itself is retained in the high security area (repository). This memory area is designed in such a manner that during operation only data is added, but not changed (WORM). You should regularly back up this area.
4. This chapter describes an alternative procedure of backing up data with disk shadow to create a shadow copy and Robo copy which performs the actual backup.

8.1 Backing up the configuration

Archive configuration must be backed up after installation and after administrative activities. Backup includes the following files:

Basic configuration of the Apache http server

```
[INSTDIR]\http\conf\httpd.conf
```

Certificate and key when using SSL encryption/https

```
[INSTDIR]\http\conf\ssl\cert.pem
[INSTDIR]\http\conf\ssl\key.pem
```

Linking the archive to the Web service

```
[INSTDIR]\config\eas.conf
```

File including trusted users and their passwords

```
[INSTDIR]\config\users.txt
```

Moreover, you need to back up a number of data files per archive. This is illustrated as an example using the store1 archive and the sample configuration from chapter 5.1.

Archive configuration

```
[INSTDIR]\config\store1.ini
```

Private and public key for signature

```
[EAS_WORM]\store1\KEYS\eas.pri-sec.dsc
[EAS_WORM]\store1\KEYS\eas.pub-sec.dsc
```

Key for aes encryption of archive data (optional)

```
[INSTDIR]\config\encryption\key
```

Backup versions of these files should be protected from third-party access.

8.2 Backing up the management data

The management data is required for production mode of the archive component. Backup during operation is difficult because this data is subject to constant change. It is therefore recommended that you hold the memory area containing the management data (EASY_RW in the configuration) redundant to protect yourself against possible hardware defects. The simplest solution is using a RAID 1 array.

Registry and search index of the archive can be recovered from the archive repository. This procedure is described in chapter 9 ("*Disaster Recovery*").

Policies and access protocols cannot be recovered; you therefore need to include them in a backup concept with minimum daily backup.

Policies control access to archive parts; they determine retention. These policies can be changed during the lifetime of an archive, which is why they were not stored in the repository's high security area. The directory for policies is defined in the archive configuration per archive, using the `policyPath` setting. This would mean the following for the `store1.ini` sample used in this documentation:

Backing up the directory containing the policies

```
[EAS_RW]\store1\POLICIES
```

The user accesses are logged in the so-called audit trail. See section 5.1.3 on how to configure it. This log cannot be reconstructed from the archived data either, which is why this directory must be included in the backup concept.

This would mean the following for the `store1.ini` sample used in this documentation:

Backing up the directory containing the access protocols

```
[EAS_RW]\store1\AUDIT
```

8.3 Backing up the archive repository

Security and backup of archived data is, of course, the most important issue. This data is stored per archive in a high security area, i.e. the repository. EAS is designed in such a manner that this is used as a WORM (Write Once Read Many) disk.

Specific storage systems are best suited to back up this area. These systems have an "inbuilt" and certified backup system.

At least this area should be secure against hardware defects. The use of a RAID 4 array is recommended.

Differential backup can be used to back up the repository to another disk.

8.4 Best Practice

In addition to the above guide saying which data of a store to back up, this chapter is to give information on how and by which means backing up the stores can be performed. This is not to say, though, there are no alternatives to backup. The focus is, above all, on what can be implemented using Windows tools. If specific backup programs are available, then their use is recommended; however, this allows implementing the same procedure as described below – often even more conveniently.

The two Windows system programs are used here

- Diskshadow (available from Windows Server 2008), and
- Robocopy (available from Windows Vista).

Diskshadow[3] is a system program which allows creating a so-called shadow copy from a partition or a directory in the file system. This is a virtual copy of the data region which conserves the state of the data region at the time of creating the shadow copy.

From this shadow copy you can now perform backup, and you can continue working on the original data range for the duration of backup. This becomes possible in that all changes in the original data region are not transmitted to the shadow copy.

[3] http://technet.microsoft.com/en-us/library/cc772172(v=ws.10).aspx

This does not mean that the data is physically copied. The system merely remembers which of the files in the data region belong to the shadow copy in which state, and which ones do not. Therefore, creating a shadow copy does not take much time, but only seconds or up to a few minutes.

The advantage of this is that the data region can be backed up in a consistent state without having to stop running the archive server for the duration of the backup operation. On the contrary, you can continue working with the archive server after creating the shadow copy.

The following example shows two scripts responsible for backing up the "DOCUMENTS_DATA" folder in which the EAS_RW and EAS_WORM regions are located.

The first of the two scripts is the script for Diskshadow:

```
#DiskshadowSampleCreate.txt
set verbose on
set context persistent
begin backup
add volume c: alias shadow_c
create
expose %shadow_c% p:
end backup
```

This script can then be executed on the command line as follows:

```
Diskshadow /s DiskshadowSampleCreate.txt
```

The shadow copy is available after the script has been executed. The metadata of the shadow copy is stored in a CAB file located in the current directory. In this example, it is irrelevant; however, in more complex scenarios, having this file can be useful. For this, we refer to the Diskshadow documentation.

The shadow copy is created through the following line:

```
add volume c: alias shadow_c
```

Here a shadow copy is created for Drive C, which contains DOCUMENTS_DATA. This shadow copy is then made available under Drive P through the

```
expose %shadow_c% p:
```

command. From that time onwards you can restart the archive server and continue with your work.

To perform actual backup, you now have to perform physical copying. This, however, is performed from the shadow copy on Drive P, and can be performed, for example, using Robocopy:

```
robocopy "P:\DOCUMENTS_DATA" "Z:\DOCUMENTS_DATA_BACKUP" /MIR /E /PURGE /V
/NP /R:10 /W:30 /XD Repository
```

Robocopy[4] is a tool deployed by Microsoft which has been optimized for creating file copies. Here the DOCUMENTS_DATA directory of the shadow copy is copied from Drive P to Drive Z. The switches here mean

- `/MIR`: Copying the entire directory structure
- `/PURGE`: Deleting files from backup that no longer exist
- `/V`: Log the processing state
- `/NP`: Only the number of copied files are to be output
- `/R`: Number of retries in case copying fails
- `/W`: Wait time between retries, in seconds
- `/XD`: Excluding directories from backup

Because Robocopy copies the shadow copies, all changes to DOCUMENTS_DATA remain invisible to Robocopy, i.e., new files or changes to existing files are not copied, which guarantees consistent backup of the stores.

In this example, the complete data region of the stores is backed up, i.e. both the WORM and the RW region. This may be useful because restoring the registry and the index can become unnecessary in case of recovery. Depending on the specific requirements, backup can be further limited, so only the WORM region is backed up, for example.

The shadow copy can then be deleted, which can be carried out by executing the following script:

```
#DiskshadowSampleDelete.txt
set verbose on
set metadata c:\example.cab
set context persistent
delete shadows exposed p:
exit
```

[4] http://technet.microsoft.com/en-us/library/cc733145(v=ws.10).aspx

9. Disaster Recovery

This chapter deals with recovering an executable archive version. The following requirements must be met to perform recovery:

- The archive data (repository) is backed up and available.
- The configuration has been backed up or can be backed up subsequently.

The scenarios in which a recovery becomes necessary are versatile.

To run EAS, registry, index and the repository must be in synchronous state. If this is not the case (e.g. owing to hardware defect, loss of data or a corrupted index), you will have to perform recovery.

The starting point for such a recovery is always the repository from which the registry and then the index are initially rebuilt. The recovery process is performed in three steps.

9.1 Step 1: Recovering the repository and the configuration

The repository and the archive configuration must be put into their recent consistent backup state. You need to stop the Windows service for EAS first. Availability of such a state is guaranteed by adhering to a backup policy.

9.2 Step 2: Restoring the registry

The archive's registry is restored first. To do this, all data files of the archive are collected and entered in the new registry that has previously been automatically created. Below the `store1` sample archive known from the previous chapters is used. You need to execute the following command via the command line using administrator privileges:

```
[C:\Program Files\eas\bin] eas.exe --crawl -c ..\config\store1.ini
```

This process may take several hours for large archives. During that time the Windows service for EAS may not be started. Prior to execution, you have to make sure that the folder structure in the RW directory is correct. It must contain the folders as specified in the store's ini file, usually AUDIT, INDEX, POLICIES, REGISTRY, and SPOOL; otherwise, there will be a cancellation and the registry and index will not be restored.

Optional switches for controlling restore

The `--crawl` switch can be combined with the `--registry-batch-size` switch. This specifies how many entries are to be collected and cached before

entries are written in the registry. High values may increase performance, but they decrease failover because the entries are initially held in RAM.

```
eas.exe [...] --crawl --registry-batch-size 100000
```

Moreover, `--crawl` can be combined with the `--time-period-begin` and `--time-period-end` switches. In this case, only the DOCUMENTS files from the specified period are collected during *crawling*.

If you specify only the start of a period but not the end, the date 12/31/9999 is the upper bound. In the reverse case, the date 1/1/1970 is the lower bound (see also chapter 9.3).

The remaining DOCUMENTS files can later be added to the registry. To do this, you have to call `--crawl` again using the `--incremental` switch:

```
eas.exe [...] --crawl --incremental
```

9.3 Step 3: Recovering the index

The next step is to reindex the archived files and attachments registered in Step 2, with the search index being rebuilt. You need to execute the following command for the familiar `store1` sample archive via the command line using administrator privileges:

```
[C:\Program Files\eas\bin] eas.exe --rebuild-index -c ..\config\store1.ini
```

The duration of this process depends on the amount of data in the archive and on the complexity of the attachments to be indexed (e.g. TIFF with an OCR filter). This might take hours, or even days.

Re-indexing only works when the EAS service is not running because the index is deleted completely during the complete re-indexing operation. Alternately, you can reset the index when the service is inactive, and then update the index with the active EAS service.

```
[C:\Program Files\eas\bin] eas.exe --reset-index -c ..\config\store1.ini

(Start EAS service)

[C:\Program Files\eas\bin] eas.exe --update-index -c ..\config\store1.ini
```

> ***Important note!***
>
> *You cannot perform parallel index tuning for the duration of re-indexing. This can thwart the re-indexing operation or even block it in such a manner that index update runs into a timeout when index tuning takes a longer time.*

If an index is broken and can no longer be opened, the `--force-reset-index` switch can be used to force resetting the index.

Optional switches for controlling restore

The `--no-records` switch suppresses indexing DOCUMENTS files. This parameter must be used in combination with the `--update-index`, `--rebuild-index` and `--restore` switches.

The `--no-attachments` switch suppresses indexing attachments. You have to use this also in combination with the `--update-index`, `--rebuild-index` and `--restore` switches.

When restoring an index, the time during which the DOCUMENTS files are to be restored can be limited. The beginning is specified using the call parameter `--time-period-begin` ; the end using `--time-period-end` .

The time must be specified according to ISO 8601, e.g. like

```
--time-period-begin 2012-08-28T9:30
```

If you specify only the start of a period but not the end, the date 12/31/9999 is the upper bound. In the reverse case, the date 1/1/1970 is the lower bound.

Updating the index can also be automatically stopped at a specific time. To do this, you need to specify the call parameter `--stop-at-time` including the time according to ISO 8601.

```
eas.exe [...] --update-index --stop-at-time 2012-11-30T18:00
```

The `--only-newest-version` switch, which can be used for rebuilding the index, ensures that only the newest version of a DOCUMENTS file is indexed.

```
eas.exe [...] --update-index --only-newest-version
```

As with registry restore using `--crawl` , the `--registry-batch-size` switch can be used for restore. Analogous to `--registry-batch-size` , the `--index-batch-size` switch is also available.

```
eas.exe [...] --update-index --index-batch-size 100000
```

Additionally, the index can be exclusively opened while restoring. The `--exclusive-index-access` switch (`-x`) is available for this purpose. This makes repeated opening and closing unnecessary. However, working side by side in the index is no longer possible because the index is locked all the time.

10. System Update

If a new EAS version should be loaded, this will usually be performed by replacing the binary files in the `bin` folder. If the Apache server is also to be updated as part of the update, you need to replace the entire `httpd` folder. Inasmuch as you have made any changes, the `httpd\conf\httpd.conf` file should be saved to obtain the changes.

Moreover, it is recommended that you back up the following data files prior to an update:

- The configuration files of all created archives in the `ini` folder and,
- where any changes have been made, the `config\eas.conf` file and, as already mentioned, the `httpd\conf\httpd.con` file.

Actual and detailed instruction and notes as to how to update will come with every updated software package.

11. Appendix: New Functions and Switches

The archive is continually developed, which also successively expands the function scope. To guarantee clear representation of new functions, these are grouped by the respective archive version – listed in this chapter.

11.1 New switches in version 1.0.1

No new switches available.

11.2 New switches in version 1.0.2

11.2.1 Section [index]

New switch to control indexing:

- `compoundFormat`
- `nonTokenChars`

Described in section 5.1.4.

11.2.2 Section [registry]

The following switches for configuring the Sqlite3 registry are new:

- `sqlite3_pagesize`
- `sqlite3_cachesize`
- `sqlite3_journalmode`

The switches are described in chapter 5.1.5.

11.2.3 New section [performanceMonitor]

This new section allows performance analyses that allow measuring how long individual archiving steps take.

This section is described in chapter 5.1.9.

11.3 New switches in version 1.0.3

11.3.1 Section [index]

The switches to control indexing are new:

- `clucene_maxSegmentsMergeSize`
- `clucene_minSegmentsMergeSize`
- `clucene_maxOptimizeSegments`

The switches are described in chapter 5.1.4.

11.3.2 Section [registry]

The registry can now be stored in an MS SQL Server database. Important: The `type=sqlserver` parameter must be set.

The following switches are available here for configuration:

- `sqlserver_name`
- `sqlserver_user`
- `sqlserver_password`

The switches are described in chapter 5.1.5.

11.3.3 New switches in the command line tool `eas.exe`

The following new parameters are available for the command line tool. You will get a complete list of all available parameters via this call: `--help` in the command line of the `eas.exe` file.

- `--no-records` (described in chapter 9.3)
- `--no-attachments` (described in chapter 9.3)
- `--max-segments <NUMBER>` (described in chapter7.5)
- `--max-merge-segment-size <NUMBER>` (described in chapter 7.5)

11.4 New switches in version 1.0.4

None.

11.5 New switches in version 1.0.5

11.5.1 Section [archive]

The switches to control archiving are new:

- `archiveBitAttribute`

This switch is described in chapter 5.1.2.

The field value based storage strategy now also allows using functions that extract the parts of a timestamp. It is described in chapter 5.1.2.

11.5.2 Section [index]

The following new parameters are available for the command line tool. You will get a complete list of all available parameters via this call: `--help` in the command line of the `eas.exe` file.

- `--time-period-begin` (described in chapter 9.3)
- `--time-period-end` (described in chapter 9.3)
- `--set-index-status` (described in chapter 7.6)

11.6 New switches in version 1.0.6

11.6.1 Section [index]

The following switches for configuring indexing are new:

- `type`

This switch is described in chapter 5.1.4.

11.6.2 Section [performanceMonitor]

Extending the action codes for the sample points. Described in chapter 5.1.9.

11.6.3 New switches in the command line tool `eas.exe`

The following new parameters are available for the command line tool. You will get a complete list of all available parameters via this call: `--help` in the command line of the `eas.exe` file.

- `--force-reset-index` (described in chapter 9.3)
- `--stop-at-time` (described in chapter 9.3)
- `--index-batch-size` (described in chapter 9.3)
- `--registry-batch-size` (described in chapter 9.3)
- `--only-newest-version` (described in chapter 9.3)
- `--exclusive-index-access` (described in chapter 9.3)

The following switches can now also be used with `--verify`:

- `--time-period-begin` (described in chapter 7.3)
- `--time-period-end` (described in chapter 7.3)

11.7 New switches in version 1.0.7

11.7.1 New section [excludedIndexFields]

The "[excludedIndexFields]" section for configuration is new. The new section is described in chapter 5.1.10.

11.8 New switches in version 1.0.8

No new switches have been added.

11.9 New switches in version 1.0.9

11.9.1 Section [archive]

- `forceNeverMaxExpire` (described in 5.1.2)

11.9.2 Section [index]

- `clucene_maxClauseCount` (described in chapter 5.1.4)
- `nonTokenChars` (described in 5.1.4)
- `nonChars` (described in 5.1.4)

11.10 New switches in version 1.0.10

No new switches have been added.

11.11 New switches in version 1.0.11

No new switches have been added.

11.12 New switches in version 1.0.12

11.12.1 Section [archive]

- `flagsPath` (described in chapter 5.1.2)

11.12.2 Section [index]

- `indexingMode` (described in chapter 5.1.4)

11.12.3 New switches in the command line tool `eas.exe`

The following new parameters are available for the command line tool. You will get a complete list of all available parameters via this call: `--help` in the command line of the `eas.exe` file.

- `--incremental` (described in chapter 9.2)
- `--migrate-registry` (described in chapter 7.7)
- `--time-period-begin` and `--time-period-end` can be used with `--crawl` (described in chapter 9.2)