
# IMiS(R)/ARChive Server Manual

Version 9.6.1606

IMAGING SYSTEMS

Imaging Systems Inc.
Brnciceva 41g
Ljubljana
Slovenia

# TABLE OF CONTENTS



 

 

 

# TABLE OF IMAGES

Table of images appearing in the manual



 

# LIST OF TABLES

Below is a list of tables appearing in the manual:



 

# 1 FOREWORD

This foreword contains a description of the contents and form of IMiS®/ARChive Server documentation. It provides useful advice on technical and content aspects of product use.

## 1.1 About the documentation

The documentation provides a description of the server architecture, individual object components, mechanisms for ensuring authenticity and security and procedures for installing, configuring and administering IMiS®/ARChive Server.

## 1.2 Target audience

The documentation is intended for experienced system administrators with a good knowledge of different versions of the Linux operating system and with experience installing, configuring and administering information systems.
Every administrator in charge of installing and maintaining IMiS®/ARChive Server must be familiar with these processes.

## 1.3 Conventions

Various styles and methods for writing important information are used in the documentation. They are summarized in the table below.

| **Types of writing** | **Purpose of use** |
|---|---|
| Normal | Basic text in the documentation |
| **Normal bold** | Chapter headings in the documentation (levels 1 to 6) |
| Normal underlined | Additional options for subchapters within an individual level |
| “Normal” | The names of functions or actions in the framework of the selection options |
| *Normal italic* | Go to another chapter |
| Monospace | Shows console commands, files, directories, ... |
| **Monospace bold** | Shows user input |

Table 1: Different styles used in the documentation

 

## 1.4 Abbreviations

The table below describes abbreviations used in the texts and illustrations:

| Abbreviation | Description |
|---|---|
| AES | Advanced Encryption Standard (an advanced encryption algorhythm) |
| AIP | Archival Information Package (A content and metadata "summary" of an entity merged into an XML document) |
| ACL | Access Control List (a list of access rights) |
| CA | Certificate Authority (a trustworthy issuer of digital certificates) |
| CIFS | Common Internet File System (a communications protocol for data, printing and other services on the network) |
| CRL | Certificate Revocation List (a list of revoked digital certificates) |
| CRM | Customer Relationship Management (a system for customer relationship management) |
| DMS | Document Management System (a system for document management) |
| EML | Email message (format for saving email messages) |
| ERP | Enterprise Resource Planning (a business information system) |
| ERS | Evidence Record Syntax (form of data for ensuring the long-term integrity of timestamps) |
| FIPS | Federal Information Processing Standard (an information processing standard) |
| HA | High Availability (a system property - high availability) |
| HMAC | Hash-based Message Authentication Code (an authentication method that includes a hashing function intended for checking the integrity of the data and authentication) |
| HSM | Hierarchical Storage Management (an object-storage concept - hierarchical document archiving) |
| ISDM | Information System for Document Management |

 

| Abbreviation | Description |
|---|---|
| LTANS | Long-Term Archive and Notary Services<br>(a group of technical procedures and functionalities for ensuring archiving over a longer period of time; also the name of the group that sets these standards) |
| MIME | Multipurpose Internet Mail Extensions |
| NAS | Network Attached Storage |
| NAT | Network Address Translation<br>(a translation of network addresses, a process for hiding private addresses) |
| OCSP | Online Certificate Status Protocol<br>(an online protocol for checking the validity status of digital certificates) |
| PDF/A | Portable Document Format<br>(a format for long-term document storage - .pdf) |
| POSIX | Portable Operating System Interface<br>(a standard interface between the software application and operating system) |
| RAID 5 | Redundant Array of Independent Disks<br>(a system for using and organizing disk drives) |
| RFC | Request for Comments<br>(a technical and organizational document, specification or public document intended for the exchange of opinions on the topic described) |
| RHEL | Red Hat Enterprise Linux<br>(a Linux server platform made by RedHat) |
| RPM | RedHat Package Manager<br>(an application for managing installed software in the Linux RedHat platform; also the software installation package for this management application) |
| SCSI/SAS | Serial Attached Small Computer System Interface<br>(a standard disk drive interface) |
| RSA | Ronald **R**ivest, Adi **S**hamir, Leonard **A**dleman<br>(a public-key encryption algorithm) |
| SLES | SUSE Linux Enterprise Server<br>(a Linux server platform made by Novell) |
| SRP | Secure Remote Password<br>(an encryption protocol for safe user authentication) |
| SHA | Secure Hash Algorithm<br>(algorithms for calculating a content hash) |
| TCP/IP | Transmission Control Protocol / Internet Protocol<br>(a family of network protocols) |

| Abbreviation | Description |
|---|---|

 

| | |
|---|---|
| TSA | Time Stamp Authority<br>(an agency for issuing certified timestamps) |
| TIFF | Tagged Image File Format<br>(a format for long-term document storage) |
| URI | Uniform Resource Identifier<br>(a uniform resource identifier that prescribes an algorithm for converting an AIP to a normal format) |
| X.509 | (the ITU-T standard for public key infrastructure use) |
| XML | Extensible Markup Language<br>(a markup language for hierarchically structuring data in the form of text files) |
| XMLDSIG | XML Signature<br>(a specification for XML records for digital signatures) |
| XSD | XML Schema Definition<br>(a W3C recommendation for defining the structure of XML documents) |
| W3C | World Wide Web Consortium<br>(a standardization body for suitable internet techniques) |
| WORM | Write Once Read Many<br>(a principle of object storage on an archive server) |
| Container | A generic element for data storage |

Table 2: Abbreviation usage in the documentation

The table below describes terms used in the texts and illustrations:

| **Term** | **Description** |
|---|---|
| IMiS®/ARChive Server | IMiS®/ARChive Storage Server<br>(archive server for object storage) |
| IMiS®/Scan | IMiS®/Scan client<br>(IMiS® client for scanned paper documents) |
| IMiS®/Storage Connector | IMiS®/Storage Connector interface<br>(interface for transferring archived objects between application and archive server) |
| IMiS®/View | IMiS®/View client<br>(IMiS® client for viewing scanned documents) |

Table 3: List of terms used in the documentation

 

# 2 INTRODUCTION

## 2.1 Presentation

IMiS®/ARChive Server is a software module for the safe long-term storage of all types of objects of electronic origin and objects digitized using various digitization processes (such as scanning).

It is a comprehensive solution for ensuring the safe, long-term storage of documents, and as such provides for sustainability, integrity, orderliness, proof of origin of records and access to records for the entire duration of their storage. It is scalable, and a practically unlimited quantity of binary documents can be stored. The delivery and display speed of the archived objects is practically not dependent on the size of the archive.

Technological measures are used to prevent the deletion or alteration of archived records.

Access rights can be managed to ensure safe access at any time and from any location.

Suitable proofs are used to ensure the authenticity of the archived content (hashes, digital signatures with digital certificates, timestamps).

The structure, creation and verification of these proofs are conducted in accordance with the ERS standard, the most sophisticated standard available. The audit trail records all access, queries and changes in IMiS®/ARChive Server.

An electronic archive presents a number of advantages over a physical archive.

Records archived in electronic form are easier to manage and essentially more efficient.

Just searching for records in the electronic archive is incomparably faster.

A simple search query can be used to access records in an instant. This would be much more difficult with a physical archive. The use of advanced technologies (OCR, FTS) greatly contributes to enhanced efficiency, as it enables searching by full document text.

Access to the archived content is available simultaneously to an unlimited number of users (provided they have rights) in different locations and in different applications. The time between when a need for a particular content arises and access to this information is essentially shorter than in the case of physical records.

The storage and administration costs of physical records increase with the amount of records stored, and quickly exceed the costs of electronic archiving.

Physical records can also be damaged, and this can result in the irreplaceable loss of key information. Unlike electronically archived content, physical records can be misplaced, lost or even intentionally misappropriated.

 

IMiS®/ARChive Server can function as a standalone electronic archive, or the user may use the interface to connect to different application servers.
Because of the highly powerful connection modules, third-party applications can perform all archiving processes and archive entity life cycle processes, manage access rights, conduct searches using metadata and full text, etc.
The archive system architecture is flexible and enables a large number of archive servers and a wide range of roles in the hierarchy to be set up.

## 2.2 Versions and identifiers

The marking of versions of the IMiS®/ARChive Server software module is based on a sequential scheme with 4 separate numeric identifiers (MAJOR, MINOR, RELEASE, BUILD) and an end identifier for the target processor architecture (ARCHITECTURE) (Linux standard).
This is the system most widely used throughout the world.
Additional product specific attributes are added to the name of the RPM installation file.
These consist of a unique archive identifier (ARCHIVEID), an identifier of the databases contained (DATABASE) and an identifier of the installation platform (PLATFORM). These are not recorded in the RPM database. They are available only to make it easier to distinguish and find the installation file:
*imisarc.MAJOR.MINOR.RELEASE-BUILD.ARCHIVEID.DATABASE.PLATFORM.ARCHITECURE.rpm*

*Example: imisarc.9.6.1606-600.0001.rdm.el4.i386.rpm*

The scheme is therefore made up of the name of the IMiS® module (“imisarc”) and the following elements:

- MAJOR: The identifier in the MAJOR position signifies the main/major version of the product. It is arbitrary and changes with regard to the volume of the changes and functionalities introduced. The identifier in this position changes the least.
  In the event that it is changed, it signifies a major difference in the product compared to the previously issued version (with a lower MAJOR version).
  This identifier has a range of values from 1-n; it is continuous and can only increase.

 

- MINOR: The identifier in the MINOR position signifies a minor version of the product. It is arbitrary and changes with regard to the volume of the changes, functionalities and fixes introduced. This identifier frequently changes and signifies smaller changes and fixes in the framework of the same product generation, which is marked with the MAJOR version. Its range of values is from 1-n; it is not continuous and with each change to the MAJOR version it reverts to the base value (1).
- RELEASE: Unlike the arbitrary range of values used throughout the world, in our product, this identifier is specific, as it signifies the chronological component of the product release using the "YYMM" format. MM signifies the month the product was released (values from 1 to 12) and YY signifies the last two digits in the year;

  *Example: The product release for June 2016, as shown in the RELEASE identifier, is 1606.*

- BUILD: The identifier in this position signifies the unique serial number of the product build. These values never repeat. In the case of a minimal change to the product within the same month, this identifier may be changed; in this case, all other identifiers remain the same. Its range of values is from 1 to n; it is not continuous and can only increase.
- ARCHIVE ID: Every instance of the installed product is assigned a unique identifier. This identifier is also used in the product security segment and in object identifiers. RPM files are delivered to clients. These files are non-transferable and are intended only for a concrete instance of the product. Its range of values is from 0001 to nnnn.
- DATABASE: This identifier signifies the database with which the product is distributed. The range of values at the time of publication of the documentation is BDB and RDM.
- PLATFORM: This identifier signifies the type of installation platform the installation package is intended for. The range of values sles8, el3 and el4 signifies the target platform. If it is a discrete platform, a suitable equivalent must be found, in line with the environment of the required system libraries.
  We help our clients make this choice, and they can also help themselves by using compatible system libraries that also enable an older (in terms of platform) product to run on newer platforms (for example, the el3 installation package + compatible libraries for the rhel3 and rhel4 platforms).

Criteria for specifying the arbitrarily determined identifiers are defined by the internal rules of the company and are subject to the process of managing change.

 

## 2.3 Functionality

- The structured archiving of contents of electronic origin or contents digitized through the scanning process.
- The storage of the metadata of archived contents using structured attributes linked in metadata schemes determined in the framework of hierarchical entity templates.
- The storage of system critical attributes that affect the life cycle of the stored contents.
- The storage of metadata and email content.
- Using metadata to manage the storage of physical records.
- The storage of the metadata of transferred records, which cannot be changed.
- Searching metadata using queries with logical and priority operators.
- Searching saved content by content elements (full text search).
- Calling up contents using external and internal identifiers and using classification codes.
- Managing a hierarchical classification scheme.
- Sorting archive contents in a classification scheme with the option of adding, deleting, editing and moving entities.
- Managing the confidentiality of records.
- Managing hierarchical access rights for the saved content, down to the level of individual metadata.
- Ensuring authenticity of the records through the creation and long-term maintenance of authenticity proof elements for the records (LTANS).
- Directory services to support user log in, management of access rights to the saved content and the expansion of the data of directory entities when used in metadata attributes.
- Recording and maintaining the audit trail of actions performed on the saved content and a secure view of the trail by authorized persons.
- Capturing and verifying the authenticity of contents by checking the validity of their embedded digital signatures.
- Ability to use both IPv4 and IPv6 protocols.
- Encryption of connections between the service and its clients.
- A highly scalable archive system with a practically unnoticeable transaction processing delay in the case of large inventories of archived contents (over 10 million entities).

 

# 3 TECHNICAL DOCUMENTATION

## 3.1 Server architecture

IMiS®/ARChive Server is the server component of the IMiS® electronic document management system. Its modular design enables a high level of adaptability to the different information systems today's largest organizations use for their operations.

At the heart of the system is its logical core, which links:

- Support services associated with external sources of information that function independently in the framework of the server instance.
- Internal logical services that provide support for the core in its decisions based on scalable business logic and that contribute to the high performance and throughput of the system.
- Network services that see to the exchange of information between server clients and the core on the one hand and that offer an integration point with back-end applications on the other. The latter exchange statistical information with the core about activity and settings.

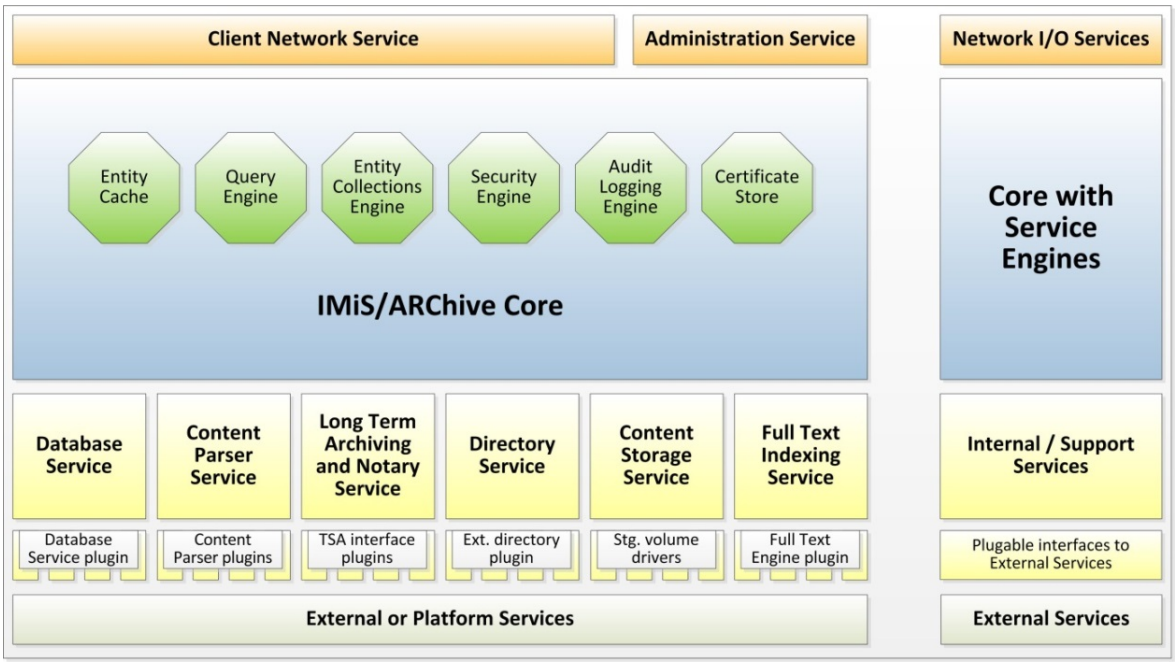


Image 1: IMiS®/ARChive Server component architecture blueprint.

 

Physically, the server represents 3 processes:

- The main process, "iarcd": a connection process that manages the support processes:
  - Receiving requests for new connections made by clients
  - Selecting logical processes for processing client connections
  - Collecting statistical data about server activity.
- Storage Volume Manager "iavol": a process for managing volumes of archived content. In its threads this process sees to optimal communication with permanent disk capacity resources, in line with their type and architecture.
  It is a support service for other IMiS®/ARChive Server processes which require these services for their operations (for example, the logical process).
- The logical process "iarcd": with its threads, this process sees to operations that users request from the archive. It is the owner of threads, which individually perform certain support services, including:
  - Content indexing by full text
  - Maintaining the authenticity of the archived contents
  - Optional synchronization of the directory with external directory service resources, ...

The logical level of the server consists of independent or mutually dependent services, which provide the core with the following:

- Support for decisions on event processing methods.
- Caching current objects which it makes sense to load to memory.
- Support for access related decisions.
- Support in maintaining the audit trail.
- The communications level of the service, which exchanges information with the clients of the service.

Servises are structured into following layers:

- External services layer, which enables the system to consume sources of information which are not part of the system.
- Embedded services layer, which provides required and optional services to the core. They enable the independent provision of certain services that aren't directly linked to the core; on the one hand they communicate with the core, and on the other they use plug-ins to access platform resources and external services.

 

- The system core with internal services which consolidates all different layers of services with built-in business logic that provides structre and rules for safe and reliable records keeping.
- The communications layer which sees to the exchange of data between the core and clients; it supports different network protocols, traffic encryption and information encryption standards.

The remainder of this chapter will present the individual services from the standpoint of their importance for server operations.

Database Service

Importance: Critical

Description: This service provides the system with data collections made up of tables and indexes necessary for operations. The service is designed abstractly, and works through the database plug-ins specified for the technology and for the database collection provider.

The following plug-ins are currently supported:

- Raima Embedded Database: An embedded database that users with rights can access locally through environment libraries for work with the database.

The system saves most information associated with the saved entities to the database. Stored content is an exception, as this is stored in the framework of a different service and in different containers.

Content Parser Service

Importance: Mandatory

Description: This system of services is used to parse contents archived by clients.

It is designed on the principle of plug-ins which process specific types of content based on the ContentType (MIMI type) of the content. Each plug-in can provide the service the following functionalities:

- Exporting content text.
- Exporting content metadata.
- Exporting embedded digital signatures.
- Exporting the digital certificates of embedded digital signatures.
- Checking the validity of embedded digital signatures.

 

The system uses these functionalities:

- Content parsing for checking content validity.
- Capturing embedded electronic signatures and their associated digital certificates.
- Support for full text indexing.

The service for plug-ins currently uses Tika and iText (Apache project) for parsing application/pdf content and a proprietary technology for parsing content in the image/tiff and text/xml formats.

Long term Archiving and Notary Service - LTANS

Importance: Not required

Description: This system optionally provides for the creation and long-term maintenance of authenticity proof elements of the stored content. The system is not essential for the operations of the archive system if the user does not require this service.

The service encompasses all closed entities, which it files in an entity inventory based on defined parameters. It provides authenticity proof elements for all entities in the inventory.

At the same time, it automatically maintains already existing authenticity proof elements and builds trees of hashed proof elements together with the new elements. It uses timestamps from different providers of trustworthy timestamps to protect these proof elements.

It obtains them via plug-ins that are specific to each timestamp provider. Additional protection from timestamp validity brakes can be provided by the use of multiple parallel timestamps from different providers of trustworthy timestamps.

By doing this the system can ensure the validity of proofs even in the rare event that the validity of timestamps is revoked before they expire.

For more information *see chapter 3.6 Authenticity*.

Directory Service

Importance: Critical

Description: This service provides support for directory services used by:

- The subsystem for the access logic of the saved content.
- The subsystem for storing metadata attribute values.

The service enables the maintenance of directory entity attributes (data on the user, the security components used to login/authenticate, data about groups to which the user belongs, etc.) and membership of directory entities (users, groups) in different groups.

 

The directory service can optionally be linked via plug-ins to external directory service resources in the sense of consolidated and centrally oriented directory services used in the framework of different systems on a distributed basis.
For more information *see chapter 3.7 Directory Services*.

Content Storage Service

Importance: Mandatory

Description: The system uses this service to permanently store content archived by clients. It is designed on the principle of plug-ins which optimally manage different, specific types of content containers. The system can therefore distinguish between different types of local content containers and different types of remote content containers.
In line with the technology, it provides the core a uniform level of content storage.
The archived content inventory is saved in line with the principles of hierarchical storage (*http://en.wikipedia.org/wiki/Hierarchical_storage_management*) on the logical level of the service for optimal use and low archiving costs. If this service is not appropriately installed, the system can function and save entity attributes, but not content.

Full Text Indexing Service

Importance: Not required

Description: This service provides the system and its users with a functionality for searching phrases in index of text extracted from archived content. It is designed on the plug-in model, which enables users and core to index new or modified content and later use the index for searching.
The system currently ships with plug-in to the Apache's Lucene runtime, which is bundled.
If the user does not require the full text search function for their archived content, this service can be disabled, as it is not critical for the functioning of the system.

Entity Cache

Importance: Critical

Description: This is an internal service is autonomously managed. The system uses a powerful cache model for all entities opened by users or other services at a certain point in time.
In line with its model, the system can internally open the entire branch of entities up to the root level and keep instances open in accordance with existing references to them.
Once the last reference to an entity is closed, the entity is removed from the cache.

 

The same is true of all its parent entities up to the root. The system thus provides a high level of responsiveness for parallel entity openings or for obtaining data from these entities.
The entire information cache model is based on the concept of the “lazy” loading of information, with the aim of ensuring a high level of responsiveness for the system. Information is lazy-loaded at first instance it is accessed by a client or a service.

If entities are changed, the attribute values of the instance for viewing and the instance for editing are loaded in memory only once and are duplicated only once the attribute value is actually changed. This ensures minimum memory use by the cache.
All the cache algorithm concepts used contribute to a system that provides a powerful, fast and effective system of access to saved content with minimum working memory use.

Query Engine

Importance: Mandatory

Description: This is an internal service with which users and the system itself search for entities on the basis of their attribute indexing values. The system provides a grammatical drive for switching the descriptive form of the query with the logical form, which is suitable for the functioning of the subsystem for searching by attribute values. Grammatical rules enable custom queries with various logical and priority operators.
The system also enables efficient searching by null attribute values.
It functions in connection with the Entity Collections Engine.

Entity Collections Engine

Importance: Critical

Description: This service enables the system to build entity collections on the basis of different types of queries. Queries can be run and information about the collected entities can be obtained from these ad hoc collections. This subsystem frequently uses client functions for hierarchical views of the tree of archived entities (all sub-entities of a parent entity, etc.) and the subsystem for searching by metadata, which builds ad hoc entity collections on the basis of a specific query.
Because entity collections can be extensive and because multiple collections can be opened simultaneously, using memory resources available to the platform wisely is crucial to the system integrity as it minimizes the effect on the amount of working memory used.

 

Security Engine

Importance: Critical

Description: Each time an entity is opened the system creates an instance for the entity if the entity is not already present in the cache. If it is not already in the cache, the saved access rules, which hierarchically enable a powerful system of determining access rights using an access control list (ACL) model, are loaded in the instance.
Rights are inherited from parent entities, and can also be replaced with explicit rights.
The rules for calculating rights are separate from the core, which is why we speak of a subsystem. The rules cannot be affected or managed, and only a limited range of settings - which do not essentially affect information protection - is available.

Audit Logging Engine

Importance: Not required

Description: This subsystem enables the core to accurately log events on the archived content in real time. These events are a part of the transactions of the operations themselves.
Audit logging on database layer can sometimes be dispersed, superfluous and hard to interprete. IMiS®/ARChive Server implements its own audit logging sub system which tracks every event that is triggered in a logically complete form, which is comprehensive, consolidated and easy to interprete by authorized personnel.
The level of audit trail logging and the volume of captured data can be managed, but any changes made to the settings will be logged in the audit trail.
If the user does not require the audit trail logging function, it can be disabled.

Certificate Store

Importance: Mandatory

Description: This subsystem enables services associated with digital certificates for the system core. These are services used by the core and/or services linked to the core.
This service saves digital certificates from trustworthy digital certificate issuers and provides logical support for verifying the validity of the different digital certificates the system encounters in its work.
This subsystem provides functionalities for obtaining data about revoked digital certificates using CRL and OSCP technologies. If the digital certificate store is empty and does not contain digital certificates from trustworthy digital certificate issuers, its operations - chekcing the validity of digital certificates - will not work.

 

Client Network Service

Importance: Critical

Description: This subsystem enables the system core to safely exchange information with its clients. An asynchronous network model is used, which enables high permeability and efficient operations of the service. Encryption of traffic between the service and clients protects the transferred information from unauthorized access. The service enables connectivity via multiple configurable TCP/IP ports and IPv4 and IPv6 protocols.

## 3.2 Attribute

In a world of enormous quantities of individual pieces of information that are interconnected or unconnected, successful storage and efficient access urgently depend on logical descriptions and links between this information.

This is accomplished using so-called metadata, which abstractly represents “information about information” or “data about data” that is the subject of storage activities.

To establish and maintain, in the long term, the structure and normalization of individual metadata, the metadata must be rounded out and rules must be written for them.

In IMiS®/ARChive Server this is achieved using the Attribute concept,

within which groups are assigned to an Entity *(see chapter 3.3 Entity)*.

An attribute is the basic cell or container of metadata.

It assigns rules and a framework for the entry, maintenance and storage of the metadata values of an entity. An attribute can also be described as a metadata model to which metadata values must conform if we would like to save them in an attribute.

There are multiple types of attributes that primarily determine the type of value that can be saved in the attribute. Attributes also contain control parameters that determine the range and form of values and possibilities for searches by these values.

Once they have been determined, attributes can be linked to form so-called metadata or attribute schemes which are merged to create entity templates *(see chapter 3.3.4 Templates)*. Links between attributes and templates determine the control parameters specific to a link between an attribute and an entity.

 

These include:

- Required.
- The possibility of containing multiple values.
- Cannot be changed.
- The form and range of a value specific to an attribute in the framework of an entity type.
- Other parameters.

An attribute is the key component and pillar of an effective electronic archive, as it describes the saved information (content) in a normalized and enables users to access the content when the body of content does not enable direct access. The reason for this can be the nature of the content (an audio or visual record) or its extent.

### 3.2.1 Types

Code: 1

Title: DirectoryEntity

Description: The identifier of an entity from the directory.

This data can be used to obtain the data of the entity from the directory.

Range of values: A registered entity in a directory.

References: /

Code: 2

Title: Bool

Description: A logical or bit value used in logical and Boolean algebra.

It signifies a true (positive) or false (negative) state.

Range of values: True, False or 1, 0

Reference: *http://en.wikipedia.org/wiki/Boolean_data_type*

Code: 3

Name: Int8

Description: A signed 8-bit integer.

Range of values: Min: -128, Max: 127.

Reference: *http://en.wikipedia.org/wiki/Character_%28computing%29*

 

Code: 4

Name: UInt8

Description: An unsigned 8-bit integer.

Range of values: Min: 0, Max: 255.

Reference: *http://en.wikipedia.org/wiki/Byte*

Code: 5

Name: Int16

Description: A signed 16-bit integer.

Range of values: Min: -32768, Max: 32767.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Short_integer*

Code: 6

Name: UInt16

Description: An unsigned 16-bit integer.

Range of values: Min: 0, Max: 65535.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Short_integer*

Code: 7

Name: Int32

Description: A signed 32-bit integer.

Range of values: Min: -2147483648, Max: 2147483647.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Long_integer*

Code: 8

Name: UInt32

Description: An unsigned 32-bit integer.

Range of values: Min: 0, Max: 4294967295.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Long_integer*

Code: 9

Name: Int64

Description: A signed 64-bit integer.

Range of values: Min: -9223372036854775808, Max: 9223372036854775807.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Long_integer*

 

Code: 10

Name: UInt64

Description: An unsigned 64-bit integer.

Range of values: Min: 0, Max: 18446744073709551615.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Long_integer*

Code: 11

Name: Int128

Description: A signed 128-bit integer.

Range of values: Min: -170141183460469231731687303715884105728,

Max: 170141183460469231731687303715884105727.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Long_integer*

Code: 12

Name: UInt128

Description: An unsigned 128-bit integer.

Range of values: Min: 0, Max: 340282366920938463463374607431768211455.

Reference: *http://en.wikipedia.org/wiki/Integer_%28computing%29#Long_integer*

Code: 13

Name: Double

Description: A real (rational) number in a floating point with a double precision.

It is not suitable for values where the absolute accuracy is required

(for example, bank transactions); in these cases the Decimal type is more suitable.

Range of values: / (dynamic range)

Reference: *http://en.wikipedia.org/wiki/Double-precision_floating-point_format*

Code: 14

Name: Date

Description: A date value without a time component.

Range of values: Min: 1.1.8192 B.C., Max: 31.12.8191.

Reference: *http://en.wikipedia.org/wiki/ISO_8601*

 

Code: 15

Name: Time

Description: A time value without a date component.

Range of values: All 24 hours, accurate to 1 millisecond.

Reference: *http://en.wikipedia.org/wiki/ISO_8601*

Code: 16

Name: DateTime

Description: A date value with a time component.

Range of values: Min: 1.1.8192 B.C. with a time component (see Date + Time),

Max: 31.12.8191 with a time component (see Date + Time).

Reference: *http://en.wikipedia.org/wiki/ISO_8601*

Code: 17

Name: StringMax

Description: An unlimited string of UTF-8 characters.

Range of values: / (limited by platform capacity)

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*Reference: http://en.wikipedia.org/wiki/UTF-8*

Code: 18

Name: String10

Description: A string of UTF-8 characters 10 bytes in length.

Range of values: 10 bytes of UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*Reference: http://en.wikipedia.org/wiki/UTF-8*

Code: 19

Name: String20

Description: A string of UTF-8 characters 20 bytes in length.

Range of values: 20 bytes of UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*Reference: http://en.wikipedia.org/wiki/UTF-8*

 

Code: 20

Name: String30

Description: A string of UTF-8 characters 30 bytes in length.

Range of values: 30 bytes of UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*http://en.wikipedia.org/wiki/UTF-8*

Code: 21

Name: String40

Description: A string of UTF-8 characters 40 bytes in length.

Range of values: 40 bytes of UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*Reference: http://en.wikipedia.org/wiki/UTF-8*

Code: 22

Name: String50

Description: A string of UTF-8 characters 50 bytes in length.

Range of values: 50 bytes of UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*Reference: http://en.wikipedia.org/wiki/UTF-8*

Code: 23

Name: String100

Description: A string of UTF-8 characters 100 bytes in length.

Range of values: 100 bytes of UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*Reference: http://en.wikipedia.org/wiki/UTF-8*

Code: 24

Name: String200

Description: A string of UTF-8 characters 200 bytes in length.

Range of values: 200 bytes of UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/String_%28computer_science%29*

*Reference: http://en.wikipedia.org/wiki/UTF-8*

 

Code: 31

Name: Decimal1

Description: A signed real (rational) number accurate to 1 decimal point.

Range of values: Min: -92233720368547758O7.8, Max: 9223372036854775808.7.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 32

Name: Decimal2

Description: A signed real (rational) number accurate to 2 decimal points.

Range of values: Min: -922337203685477580.78, Max: 922337203685477580.87.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 33

Name: Decimal3

Description: A signed real (rational) number accurate to 3 decimal points.

Range of values: Min: -92233720368547758.078, Max: 92233720368547758.087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 34

Name: Decimal4

Description: A signed real (rational) number accurate to 4 decimal points.

Range of values: Min: -9223372036854775.8078, Max: 9223372036854775.8087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 35

Name: Decimal5

Description: A signed real (rational) number accurate to 5 decimal points.

Range of values: Min: -922337203685477.58078, Max: 922337203685477.58087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 36

Name: Decimal6

Description: A signed real (rational) number accurate to 6 decimal points.

Range of values: Min: -92233720368547.758078, Max: 92233720368547.758087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

 

Code: 37

Name: Decimal7

Description: A signed real (rational) number accurate to 7 decimal points.

Range of values: Min: -9223372036854.7758078, Max: 9223372036854.7758087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 38

Name: Decimal8

Description: A signed real (rational) number accurate to 8 decimal points.

Range of values: Min: -922337203685.47758078, Max: 922337203685.47758087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 39

Name: Decimal9

Description: A signed real (rational) number accurate to 9 decimal points.

Range of values: Min: -92233720368.547758078, Max: 92233720368.547758087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 40

Name: Decimal10

Description: A signed real (rational) number accurate to 10 decimal points.

Range of values: Min: -9223372036.8547758078, Max: 9223372036.8547758087.

Reference: *http://en.wikipedia.org/wiki/Decimal*

Code: 41

Name: Binary

Description: An unlimited bit string with data about the content type (MIME);

it is capable of transmitting the size of the saved string.

Range of values: / (limited by platform capacity)

Reference: *http://en.wikipedia.org/wiki/Bitstring*

*Reference: http://en.wikipedia.org/wiki/Internet_media_type*

 

Code: 42

Name: File

Description: An attribute capable of storing files with: data about the content type (MIME), a description, time of creation, last changed and last accessed.
It is also capable of transmitting the size of the saved string.

Range of values: Content up to 9223372036854775807 bytes in size, description with a size of up to 4096 UTF-8 characters.

Reference: *http://en.wikipedia.org/wiki/Internet_media_type*

### 3.2.2 Parameters

Besides its type, an attribute is described by the following data:

- "Name": a name, a required unique string of UTF-8 characters up to 256 bytes of UTF-8 characters in length.
- "Description": a description, an optional string of UTF-8 characters up to 512 bytes of UTF-8 characters in length.
- "Validation expression", a string of UTF-8 characters that represent a regular expression with which the new or changed values of the attribute are checked.
  For additional information on syntax and rules see:
  *http://en.wikipedia.org/wiki/Regular_expression*.
- "Parameters": additional attribute parameters which further constrain values assigned to the attribute: "Searchable", "Unique" and "PickList".

"Searchable"

The values of this attribute become a part of an index which can be used with the search function to search for entities with assigned values. Conversely, attributes not marked as "Searchable" cannot be searched for using the search function.
On the basis of this fact, when entering settings for IMiS®/ARChive Server and its attributes, the user must decide whether or not to assign this parameter to an attribute.
Once the attribute receives a value, this parameter can no longer be changed.
This parameter can essentially affect the operations and performance of the server.
The data is saved in search trees, which can become large and slow. On the other hand, attributes with the "Non-searchable" value are only saved, meaning that their quantity does not essentially affect server performance.

 

"Unique"

Each attribute value is unique throughout the entire archive. Select this parameter if you would like to prevent a value from being entered for an attribute that is already specified by a different entity. If this rule is violated, the system will prevent the entity from being saved. It is also possible to work with the parameter even after an attribute has already been assigned values. Or it can be turned off. As long as an attribute does not have assigned values, this parameter can be managed freely.

"PickList"

Attribute values are determined in advance, which is why manual entry outside the list of allowed values is not possible. The list of allowable attributes is made when configuring IMiS®/ARChive Server.

Once the list has been assigned values, additional values can simply be added.

As long as no attribute has a value, the value can be removed from the list.

Additionally, for each value the following can be specified:

- Attributes:

  - "alias": a synonym, a user-friendly value name that clients can use to select values

  - "alias.xx_YY": a translated synonym, a user-friendly value translated into a field identified with the identifier xx_YY; clients can use this value to select values when a client is working in the field settings of xx_YY; field identifiers of xx_YY are structured in accordance with the ISO 639 standard (*http://en.wikipedia.org/wiki/ISO_639*)

  - "default": The default value of an attribute in the case of new entities; clients can use this attribute property in user interfaces to display the default values of an attribute

- Conditions: Lists the conditions under which a certain value is allowed; conditional syntax for pairs can be used, or only the value itself if you would like it to apply to an attribute for which conditions have been prescribed.

*Examples:*

- *"1:{default,alias="Opened",alias.sl_SI="Odprto"} – The attribute value "1" is identified by the alias "Opened" and translated into Slovenian as "Odprto". It is default for new entities. Conditions for when it is available are not listed.*
- *"2:{alias="Closed",alias.sl_SI="Zaprto"}:{"":"1"} – The attribute value "2" is identified by the alias "Closed" and translated into Slovenian as "Zaprto".*
  *It is available on the condition that the attribute does not have a value (a new entity) or its value is 1.*

 

### 3.2.3 Links to templates

An attribute is not independently capable of storing metadata; it is only the basis or source of the data container. The metadata value container is ultimately determined by the link established between the attribute and a template. Simply put, templates are settings on the basis of which entities are created in the archive *(see chapter 3.3.4 Templates)*.

In connection with an attribute, it is also necessary to be familiar with the concept of how templates work. Inheritance significantly affects an understanding of the metadata schemes of the executed templates. The link between a template and an attribute is the basic element of an entity's metadata scheme.

It determines the so-called contract, that is, all parameters and restrictions that all metadata values entered for an attribute must adhere to, if we would like to enter them in the attribute that contains them.

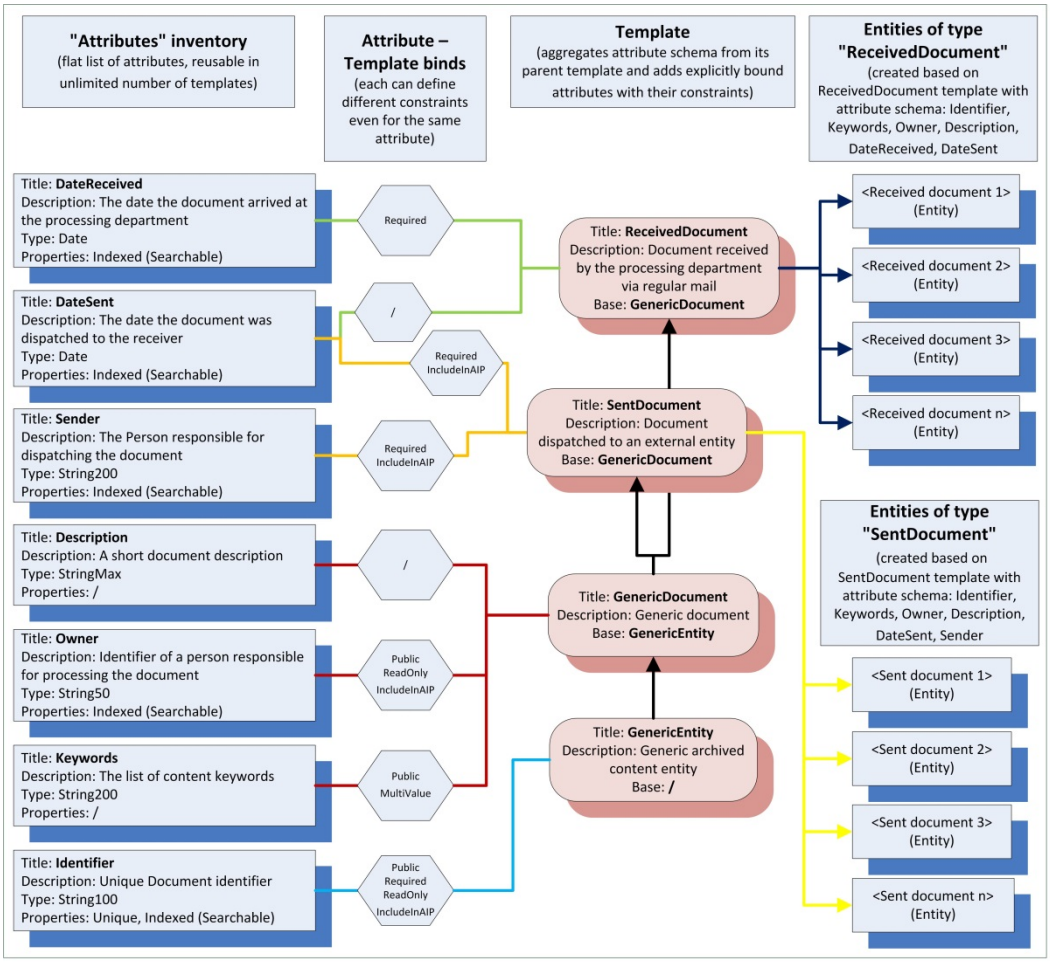


Image 2: Logical links between attributes, templates and entities

 

This model enables the same attribute to be reused in different templates, and it is also possible to determine certain attribute parameters on the level of the link.
These parameters are:

- "Validation expression": A string of UTF-8 characters that represent a regular expression used to check new or changed attribute values; for additional information on syntax and rules see: *http://en.wikipedia.org/wiki/Regular_expression*.
  This expression replaces the expression set on the level of the attribute. Checking by an expression of an attribute is therefore replaced by checking by an expression on the level of the link between the attribute and the template.
- "Sequence": The sequence of the attribute in the metadata scheme of the template.
  This is the sequence of the attribute in the scheme. The following rules must be taken into account:
  1. The scheme of each parent template has its own sequential order which does not depend on the parent template and/or any templates created from it.
  2. The attributes of the root parent template scheme are initially consolidated attribute schemes.
  3. The attributes of the scheme of the executed template from which entities are created are ultimately the consolidated attribute schemes of this template.
- "PickList values": when necessary, a range of predefined attribute values, which are determined globally, can be used. The list is saved in the context of the link between the attribute and the template. Manual entry of values outside the list of allowed values is not possible. The list of allowed values is determined when entering settings for the product.
  Values can only be added to the list once its values have been defined.
  As long as they do not have a defined value, values can also be removed.
  For more information *see Options, "PickList"*.
- "Parameters": additional attribute parameters that further define its assigned values: "Searchable", "Unique" and "PickList".

"Public"

The public attribute. The values of this attribute in the framework of an entity are of a public nature and are also accessible to directory entities (users, groups) who otherwise do not have access rights for the entity. This parameter can be managed throughout the entire life cycle of the attribute and/or the template to which the attribute is linked. It is used the first time an entity is opened by a user following the changed setting.

 

"Multivalue"

An attribute that can contain multiple values. The attribute can be assigned multiple values in the framework of a single entity. If this parameter is not specified, an attribute can be assigned exactly one value (or no value) in the framework of a single entity. This parameter can be managed throughout the entire life cycle of the attribute and/or the template to which the attribute is linked until the first entity has been created on the basis of the template.
It can then be selected at any time. It cannot be turned off. It is used the first time an entity is opened by a user following the changed setting.

"Required"

An attribute value must be present when the entity is saved. Server will deny the save if the saver of the entity does not specify at least one valid value for the attribute.
This parameter can be managed throughout the entire life cycle of the attribute and/or the template to which the attribute is linked until the first entity has been created on the basis of the template. It can then be selected at any time. It cannot be turned off.
It is used the first time an entity is opened by a user following the changed setting.

"ReadOnly"

The values of the attribute cannot be edited once it has been saved.
The attribute values can be specified the first time the entity is saved. All later attempts to edit the attribute will be rejected. This parameter can be managed throughout the entire life cycle of the attribute and/or the template to which the attribute is linked.
It is used the first time an entity is opened by a user following the changed setting.

"Inherited"

The values of the attribute inherit values from the parent hierarchy. If values are not explicitly specified in an entity, they are inherited from the first parent entity for which values are explicitly specified. Explicit values as a whole are above inherited values and have an effect on all child entities.
Inheriting a value works on the principle of references. If the value of the inherited attribute is changed in the parent of an entity, it immediately takes effect for all child entities for which explicit values have not been defined.
Attribute values are only inherited if an entity and the entity directly above it have the attribute in their attribute scheme.

 

This parameter can be managed throughout the entire life cycle of the attribute and/or the template to which the attribute is linked. It is used the first time an entity is opened by a user following the changed setting.

"AppendOnly"

The values of the attribute can only be appended to existing values. The client receives all values from the server; when writing it only accepts new values, which it adds to already existing ones.

This parameter can be managed throughout the entire life cycle of the attribute and/or the template to which the attribute is linked. It is used the first time an entity is opened by a user following the changed setting.

"IncludeInAIP"

The values of the attribute are a part of the archival information package (AIP). During the procedure for determining the authenticity of the stored records, the values of attributes with this parameter set become a part of the archive information package (*see chapter 3.6.5 AIP*).

This parameter can be managed throughout the entire life cycle of the attribute and/or the template to which the attribute is linked. The server uses this setting when a procedure for determining the authenticity and integrity of the records is begun while the records are stored.

"IsNonEmpty"

The value of this attribute can't be empty. The definition of the empty value depends on the type of attribute. Thus, the blank value for line »String10« is an empty string, and an empty value for string »File« represents an empty file (file size 0 bytes). You can manage this parameter throughout the entire lifecycle of the attribute and/or template. This applies to the attribute that is connected until the first template-based entity is created. After that it is possible to switch off the parameter at any time and never switch it on. This is taken into account when the user opens the entity for the first time after changing the setting.

*Restrictions:*

*When configuring attributes in templates with already existing entities, the following restrictions apply:*

- *When adding a new attribute, the property »Required« can't be set on »True«.*
  *If it is set on »False«, it leads to inconsistency with all existing entities as these can't have values for the new attribute.*
- *When changing an existing attribute in a template, the following restrictions apply:*

 

- *a change of attribute's property »Multivalue« from »True« to »False« is not allowed because of inconsistency with all the existing entities that have multiple values set for this attribute;*
- *a change of attribute's property »Required« from »False« to »True« is not allowed because of inconsistency with all the existing entities that don't have a value for this attribute;*
- *a change of attribute's property »Inherited« from »True« to »False« is not allowed because of inconsistency with all the existing entities that don't have an explicit value for this attribute.*

• *Deleting an attribute from a template that has existing entities is not allowed.*

*All other combinations are allowed.*

### 3.2.4 Naming

Every attribute must be named using a custom unique string of UTF-8 characters up to 256 UTF-8 characters in length. All characters are allowed, but the following restrictions should be noted:

- An attribute name cannot be blank.
- The prefixes (namespaces) “int:”, “sys:”, “imp:”, and “trf:” are reserved and their use is not allowed; attempts to create an attribute with these letter combinations will be rejected.
- Attribute names should make sense and be short.
- Prefixes or namespaces separated from the name with a “:” sign can be used; nested prefixes are allowed (for example “global:accounting:InvoiceNo”).
- The system does not have rules for naming attributes, but it is wise to assign names on the basis on an internal rulebook for work with IMiS®/ARChive Server.

It is also smart to determine an attribute description, that is, an optional string of UTF-8 characters up to 512 characters in length that should contain a logical description of what the attribute represents and which values are allowed if restrictions exist.
This data is transferred to clients that can display it in the user interface.
Localization on the level of the server is not supported.
Strings of characters can however be used as constants, which are then translated in accordance with the regional settings of clients.

 

### 3.2.5 System attributes

The operations of the archive system are greatly affected by certain attributes and their values.

These attributes are called “System attributes”; unlike attributes with settings, they are defined in the system in advance and their properties cannot be changed.

The reason for this is that their presence/absence, values and properties affect server operations, entity life cycles and processes for record storage.

System attributes are built into IMiS®/ARChive Server and cannot be managed.

They are linked to system templates from which the templates used to create entities are taken. Created entities therefore implicitly receive them in their attribute schemes.

They therefore affect entity life cycles and the presence of system metadata.

The values of some system attributes are defined by the server in the processes through which it creates, edits and saves entities. For some system attributes, the user must enter a value(s), as only in this way is it possible to successfully manage the entity life cycle.

System attributes are described in detail below.

**»sys:ExternalIds«**

Type: String100

Properties: Unique, Searchable, Public, MultiValue, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties; one entity can contain up to 65536 external identifiers.

Use: Class, Folder, Document, Review process.

Description: Unique external entity identifiers; they are used as data for access to the entity (opening); external entity is responsible for the uniqueness of the identifier, and saving is not allowed if the same external identifier has already been assigned to another entity.

**»sys:Title«**

Type: String200

Properties: Searchable, Public, Required, IsNonEmpty [ReadOnly with copies of retention policies and deleted entities].

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document, Review process, Retention policies, Retention and disposition process, Copy of retention policy, Disposition hold, Container of system entities.

 

Description: The required name (title) of the entity it describes; this can be changed throughout the entire entity life cycle. This attribute does not affect the business logic of the server in operations with entities. It is only a container of information.

**»sys:Description«**

Type: String200
Properties: Public, [ReadOnly in deleted entities and copies of retention policies].
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Class, Folder, Document, Review process, Retention policies, Retention and disposition process, Copy of retention policy, Disposition hold, Container of system entities.
Description: A non-required short description of the entity; this can be changed throughout the entire entity life cycle. If an entity is deleted, it becomes required; this attribute does not affect the business logic of the server in operations with entities. It is only a container of information.

**»sys:Content«**

Type: File
Properties: Searchable, IncludeInAIP, MultiValue, [Public in Review process, Public and ReadOnly in Copies of retention policies].
Restrictions: There are no restrictions except those pertaining to the type or properties; one entity can contain up to 65536 values.
Use: Document, Document in review process, Retention politics, Retention and disposition process, Copy of retention policy.
Description: This attribute represents the content container and is able to save the content's descriptive structures. This attribute is included in the group of attributes that are indexed; the specific feature of a File attribute is that it represents the Full Text Index.

If a system for ensuring authenticity and integrity for the duration of record storage is in place, the hashes of the values from this container become a part of the Archival Information Package (AIP). It can be changed throughout the entire entity life cycle. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:Keywords«**

Type: String30

Properties: Searchable, Public, MultiValue, Document in review process, Review process.

Restrictions: There are no restrictions except those pertaining to the type or properties; one entity can contain up to 65536 values.

Use: Class, Folder, Document.

Description: Non-required keywords that define the entity. It can be changed throughout the entire entity life cycle. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:SecurityClass«**

Type: UInt32

Properties: Searchable, Public, ReadOnly, Inherited, PickList, IsNonEmpty.

Restrictions: Valid values (predefined):

- 1, alias "Unclassified": A degree of confidentiality has not been defined for the entity.
- 2, alias "Restricted": The entity is internal in nature; only those users with a security class of "Restricted" or higher may access it.
- 3, alias "Confidential": The entity is classified; only those users with a security class of "Confidential" or higher may access it.
- 4, alias "Secret": The entity is secret; only those users with a security class of "Secret" or higher may access it.
- 5, alias "Top Secret": The entity is top secret; only those users with a security class of "Top Secret" or higher may access it.

Security classes can be set in accordance with the archive user's rulebooks for archiving.

The value 0 is invalid because it is reserved and internal, which is why it cannot be entered as a valid value.

Use: Class, Folder, Document.

Description: This system attribute restricts access to the entity. An entity to which a user with a particular security class does not have access will be hidden for that user.

The user cannot confirm its existence or perform any activity through which it would be possible to confirm its existence. Besides reading its attributes, opening, deleting, moving, etc., this also holds for creating entities under it.

The security class value is inherited by child entities if it has not been defined. Child entities can be assigned their own security class, but only up to the level of the parent entity; in other words, their security class can be the same as or lower than that of the parent entity.

 

**»sys:Status«**

Type: UInt32

Properties: Searchable, Public, Required, PickList, Inherited, ReadOnly, IsNonEmpty [Inherited property does not apply in the case of Review process].

Restrictions: Valid values:

- 1, alias "Opened": users are allowed to edit the entity and to create child entities beneath it.
- 2, alias "Closed": users are not allowed to change the entity or to create child entities.

The presence of this attribute is required in the class, in the document beneath the class and in folders.

Use: Class, Folder, Document, Retention and disposition process.

Description: This attribute represents the status of the entity. This status is reflected in the processes that are allowed or prohibited for the entity. An entity with open status can be edited and freely managed. The access level is determined by access rights.

Once an entity is assigned closed status, editing is no longer possible, regardless of access rights. If a service for ensuring authenticity and integrity for the duration of storage is enabled, closing an entity sends the signal for beginning the process for ensuring authenticity for the concrete entity and for any child entities. (*see chapter 3.3.7 Life Cycle*).

**»sys:Creator«**

Type: DirectoryEntity

Properties: Searchable, Public, Required, ReadOnly, IsNonEmpty [Inherited in the case of Container of system entities].

Restrictions: This value is defined by the server and cannot be changed.

This value always represents a value from the directory.

Use: Class, Folder, Document, Document in Review process, Retention politics, Retention and disposition process, Copy of retention policy, Disposition hold, Container of system entities.

Description: This value represents a directory entity - the user who created the entity.

This data cannot be edited and is determined when an entity is created by the server.

This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:Owner«**

Type: DirectoryEntity

Properties: Searchable, Public, IsNonEmpty.

Restrictions: The allowed values are identifiers of registred entities from the directory.

Use: Class, Folder, Document, Document in Review process, Retention and disposition process.

Description: This value represents a directory entity - the user or group responsible for the entity (owner). This can be edited throughout the entire entity life cycle.

This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:Significance«**

Type: UInt32

Properties: Searchable, Public, Inherited, PickList, IsNonEmpty.

Restrictions: Valid values:

- 1, alias “Vital”: An entity that is vitally important for the archive owner. Deleting the entity through an administrator request or in the retention and disposition process is prohibited. The entity is optionally under a special secure archiving regime.
- 2, alias “Permanent”: Deleting the entity through an administrator request or in the retention and disposition process is prohibited. This is just a warning.
  The administrator can take it into account or ignore it.
- 3, alias “Retain”: This is a warning for the person who is performing retention and disposition process. The disposition process should not be carried out on an entity with this status.
- 4, alias “Delete”: A recommendation telling the administrator to delete the entity outside of the retention and disposition process. This recommendation can be issued each time someone edits the entity. This is usually how entities that have been entered incorrectly are deleted. The deleting can also be requested by the document owner (personal data, etc.)

Use: Class, Folder, Document.

Description: This attribute represents the importance of the document to the archive owner.

 

**»sys:Opened«**

Type: DateTime

Properties: Searchable, Public, ReadOnly, IsNonEmpty.

Restrictions: This value is defined by the server and cannot be changed. It is always present with the "sys:Status" system attribute.

Use: Class, Folder, Document, Retention and disposition process.

Description: This value represents the date and time when the "sys:Status" attribute of an entity received the "Opened" value *(see chapter 3.2.5 "sys:Status")*.

This data cannot be edited and is determined by the server. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:Closed«**

Type: DateTime

Properties: Searchable, Public, ReadOnly, Inherited, IsNonEmpty [Inherited property does not apply in the Review process].

Restrictions: This value is defined by the server and cannot be changed. It is always present with the "sys:Status" system attribute.

Use: Class, Folder, Document, Retention and disposition process.

Description: This value represents the date and time when the "sys:Status" attribute of an entity received the "Closed" value. *(see chapter 3.2.5 "sys:Status")*.

This data cannot be edited and is determined by the server. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:CommitLog«**

Type: StringMax

Properties: ReadOnly, MultiValue, AppendOnly, IncludeInAIP, IsNonEmpty.

Restrictions: This value is defined by the server and cannot be changed.

Use: Class, Folder, Document, Document in Review process, Retention policies, Retention and disposition process, Copy of retention policy.

Description: This value represents an event log of automatic checking actions performed by the server when an entity is saved. Besides a text log, it also contains the digital signatures of the archived content, digital certificates, whole chains of certificates that issued the certificate with which the digital signature was created and current lists of revoked certificates from all issuers of these certificates. This data cannot be edited and is determined by the server.

 

The data in this attribute is included in the Archival Information Package (AIP) and is subject to long-term record storage in the authenticity verification process.
This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:move:Reason«**

Type: String200
Properties: Searchable, MultiValue, ReadOnly, AppendOnly, IsNonEmpty.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Class, Folder, Document.
Description: These values represent reasons for moving (reclassifying) an entity.
It is a value set by the user when an entity is moved in the classification hierarchy of the records. This value can be linked with the "sys:move:Agent", "sys:move:DateTime" and "sys:move:ClassificationCode" attributes to provide more complete information about the movement of an entity in the record classification hierarchy.
This value is entered in the metadata of the entity that is moved; it is not entered in its child entities. For more information *see chapter 3.3.7 Life Cycle*.
This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:move:Agent«**

Type: DirectoryEntity
Properties: Searchable, MultiValue, ReadOnly, AppendOnly, IsNonEmpty.
Restrictions: There are no restrictions except those pertaining to the type or properties.
These values can represent the identifiers of registered entities from the directory.
Use: Class, Folder, Document.
Description: These values represent directory entities (users) who have moved (reclassified) an entity. This value is automatically set by the server using the user session or the log-in data of the user who moved the entity.
This value can be linked with the "sys:move:Reason", "sys:move:DateTime" and "sys:move:ClassificationCode" attributes to provide more complete information about the movement of an entity in the record classification hierarchy.
This value is entered in the metadata of the entity that is moved; it is not entered in its child entities. For more information *see chapter 3.3.7 Life Cycle*. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:move:DataTime«**

Type: DateTime

Properties: Searchable, MultiValue, ReadOnly, AppendOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: This value represents the date and time when an entity was moved (reclassified). This value is automatically set by the server.

This value can be linked with the "sys:move:Reason", "sys:move:Agent" and "sys:move:ClassificationCode" attributes to provide more complete information about the movement of an entity in the record classification hierarchy. This value is entered in the metadata of the entity that is moved; it is not entered in its child entities.

For more information *see chapter 3.3.7 Life Cycle*. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:move:ClassificationCode«**

Type: String200

Properties: Searchable, MultiValue, ReadOnly, AppendOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: These values represent a classification code that is used every time the entity is moved (reclassified). This value is automatically set by the server.

This value can be linked with the "sys:move:Reason", "sys:move:Agent" and "sys:move:DateTime" attributes to provide more complete information about the movement of an entity in the record classification hierarchy. This value is entered in the metadata of the entity that is moved; it is not entered in its child entities.

For more information *see chapter 3.3.7 Life Cycle*. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:del:Reason«**

Type: String200

Properties: Searchable, Public, Required, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: This value represents the reason for dispositioning or deleting an entity. This value is set by the user when an entity is deleted. In the case of disposition, it is transferred from the review process. For more information *see chapter 3.3.7 Life Cycle*. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:del:Agent«**

Type: DirectoryEntity

Properties: Searchable, Public, Required, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties. These values always represent the identifiers of registered entities from the directory.

Use: Class, Folder, Document.

Description: This value represents the directory entity (user) who performed disposition or delete of the entity. This value is automatically set by the server using the user session or the log-in data of the user who performed the action. For more information *see chapter 3.3.7 Life Cycle*. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:del:DateTime«**

Type: DateTime

Properties: Searchable, Public, Required, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document

Description: This value represents the date and time an entity was disposed or deleted. This value is automatically set by the server. For more information *see chapter 3.3.7 Life Cycle*. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:del:ClassificationCode«**

Type: String200

Properties: Searchable, Public, Required, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: This value represents the classification code of the entity at the time of disposition or deletion. This value is automatically captured by the server. For more information *see chapter 3.3.7 Life Cycle*. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:del:Reference«**

Type: String200

Properties: Searchable, Public, ReadOnly

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: The value represents a reference to the transferred entity. This value is set by the user after successfully transferring the entity in the review process.
This attribute does not affect the business logic of the server in operations with entities;
it is only a container of information.

**»sys:scc:Reason«**

Type: String200

Properties: Searchable, MultiValue, ReadOnly, AppendOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: These values represent reasons for changing the security class of an entity.
This value is set by the user when the security class of an entity is changed.
This value can be used in conjunction with the “sys:scc:Agent”, “sys:scc:DateTime”, “sys:scc: From” and “sys:scc:To” attributes to provide more complete information about a change made to the security class of an entity. This value is entered in the metadata of the entity whose security class is changed; it is not entered in its child entities. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:scc:Agent«**

Type: DirectoryEntity

Properties: Searchable, MultiValue, ReadOnly, AppendOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties. These values always represent the identifiers of registered entities from the directory.

Use: Class, Folders, Document.

Description: These values represent directory entities (users) who change the security class of an entity. This value is automatically set by the server using the user session or the log-in data of the user who changed the security class of the entity. This value can be used in conjunction with the “sys:scc: Reason”, “sys:scc:DateTime”, “sys:scc: From” and “sys:scc:To” attributes to provide more complete information about a change made to the security class of an entity. This value is entered in the metadata of the entity whose security class is changed; it is not entered in its child entities. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:scc: DateTime«**

Type: DateTime

Properties: Searchable, MultiValue, ReadOnly, AppendOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: These values represent the data and time a change was made to the security class of an entity. This value is automatically set by the server. This value can be used in conjunction with the “sys:scc: Reason”, “sys:scc: Agent”, “sys:scc: From” and “sys:scc: To” attributes to provide more complete information about a change made to the security class of an entity. This value is entered in the metadata of the entity whose security class is changed; it is not entered in its child entities. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:scc: From«**

Type: UInt32

Properties: Searchable, MultiValue, ReadOnly, AppendOnly, PickList.

Restrictions: These values must conform to all values of the "sys:SecurityClass” that existed at any point following product installation. A value must also be defined for “0”, which is set in advance as an alias for “Unspecified”. It can also be changed.

Use: Class, Folder, Document.

 

Description: These values represent the security class of an entity prior to the security class change. This value is automatically set by the server using the value of the “sys:scc” attribute prior to the change. This value can be used in conjunction with the “sys:scc:Reason”, “sys:scc: DateTime”, “sys:scc: DateTime” and “sys:scc: To” attributes to provide more complete information about a change made to the security class of an entity.
This value is entered in the metadata of the entity whose security class is changed; it is not entered in its child entities. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:scc:To«**

Type: UInt32
Properties: Searchable, MultiValue, ReadOnly, AppendOnly, PickList.
Restrictions: These values must conform to all values of the "sys:SecurityClass” attribute that existed at any point following product installation. A value must also be defined for “0”, which is set in advance as an alias for “Unspecified”. It can also be changed.
Use: Class, Folder, Document
Description: This value represents the security class once the security class has been changed. It is automatically defined by the server using the value of the “sys:scc” attribute following a change. This value can be used in conjunction with the “sys:scc:Reason”, “sys:scc:Agent”, “sys:scc: DateTime” and “sys:scc: From” attributes to provide more complete information about a change made to the security class of an entity.
This value is entered in the metadata of the entity whose security class is changed; it is not entered in its child entities. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

### 3.2.6 Email document attributes

Some attributes and templates for email archiving are predefined in IMiS®/ARChive Server. They are intended for storing metadata about email. These attributes are predefined on the server and their properties cannot be managed.

**»sys:eml:MessageId«**

Type: String100
Properties: Searchable, ReadOnly.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Document.

 

Description: This value represents the unique identifier of a message specified by the mail server upon delivery. The value is provided by the client, and is usually extracted from the email itself, although the accuracy of the information ultimately depends on the client.
The value represents the value from the “message-id” attribute of the email, in line with the RFC 2822 specification. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:eml:Date«**

Type: DateTime
Properties: Searchable, Required, ReadOnly, IsNonEmpty.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Document.
Description: This value represents the date and time the message was sent. In line with the specifications, this is the moment at which the sender decided that the message was suitable for sending and began the process for sending the message. The value is provided by the client, and is usually extracted from the email itself, although the accuracy of the information ultimately depends on the client. The value represents the value from the “orig-date” attribute of the email, in line with the RFC 2822 specification. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:eml:From«**

Type: String200
Properties: Searchable, Required, ReadOnly, IsNonEmpty.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Document.
Description: This value represents the valid email address of the sender of the email.
The value is provided by the client, and is usually extracted from the email itself, although the accuracy of the information ultimately depends on the client. The value represents the value from the “from” attribute of the email, in line with the RFC 2822 specification.
This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:eml:To«**

Type: String200

Properties: Searchable, MultiValue, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Document.

Description: These values represent the valid email addresses of the recipients of the email. The value is provided by the client, and is usually extracted from the email, although the accuracy of the information ultimately depends on the client. The value represents the value from the “to” attribute of the email, in line with the RFC 2822 specification. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:eml:ToCC«**

Type: String200

Properties: Searchable, MultiValue, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Document.

Description: These values represent the valid email addresses of the recipients in the cc field of the email. The value is provided by the client, and is usually extracted from the email, although the accuracy of the information ultimately depends on the client. The value represents the value from the “cc” attribute of the email, in line with the RFC 2822 specification.
This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:eml:ToBCC«**

Type: String200

Properties: Searchable, MultiValue, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Document.

Description: These values represent the valid email addresses of undisclosed recipients in the bcc field of an email message. The value is provided by the client, and is usually extracted from the email, although the accuracy of the information ultimately depends on the client.
The value represents the value from the “bcc” attribute of the email, in line with the RFC 2822 specification. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:eml:Subject«**

Type: String200

Properties: Searchable, ReadOnly.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Document.

Description: This value represents the title of the subject of the email. The value is provided by the client, and is usually extracted from the email itself, although the accuracy of the information ultimately depends on the client. The value represents the value from the "subject" attribute of the email, in line with the RFC 2822 specification. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:eml:Priority«**

Type: String20

Properties: Searchable, PickList, ReadOnly, IsNonEmpty.

Restrictions: Valid values:

- Normal: The normal priority for delivery of the email (RFC 1327: "normal").
- Low: The low priority for delivery of the email (RFC 1327: "non-urgent").
- High: The high priority for delivery of the email (RFC 1327: "urgent").

Use: Document.

Description: This value represents the delivery and processing priority of the email.
The value is provided by the client, and is usually extracted from the email itself, although the accuracy of the information ultimately depends on the client.
This attribute does not affect the business logic of the server in operations with entities;
it is only a container of information.

**»sys:eml:Signed«**

Type: Bool

Properties: Searchable, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Document.

Description: This value represents data on whether or not the email is digitally signed.
The value is provided by the client, and is usually extracted from the email itself, although the accuracy of the information ultimately depends on the client.
This attribute does not affect the business logic of the server in operations with entities;
it is only a container of information.

 

### 3.2.7 Physical record management attributes

If the digital archive is used to store digitalized content/records or even just the metadata of such content/records, an efficient system of linking the digital portion of the records with their physical counterparts must be established. For this purpose, the system provides physical record management attributes.

Maintaining current values in these attributes enables the simple, accurate and traceable management of physical records. The custodians of the physical records are responsible for maintaining these values. They do this by entering any borrows or transfers in and out of the system. Each change of an attribute of a physical record is entered in the audit trail of the digital entity describing the physical entity. Traceability is thus ensured.

All physical record management attributes are registered in the "prm" space in the title.

These attributes cannot be edited or assigned additional values. They are linked to system templates from which the templates capable of archiving folders and/or documents are derived. Except for their effect on the audit trail, physical record management attributes do not affect the business logic of the server in its operations with entities.

They are only containers of information.

**»sys:prm:Identifier«**

Type: String100

Properties: Searchable, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Folder, Document.

Description: This value represents the unique identifier of the physical record.

It uniformly designates the underlying physical record. This value can be edited throughout the entire entity life cycle. It is set by the custodian of the physical record, who must be given the ability to edit the entity's metadata. This data is not required. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:prm:Description«**

Type: String200

Properties: Searchable.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Folder, Document.

 

Description: This value represents a description of the physical record. This data is used to describe, to the greatest possible extent, the record and its format, physical appearance, length, etc. This value can be edited throughout the entire entity life cycle. It is set by the custodian of the physical record, who must be given the ability to edit the entity's metadata. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:prm:Status«**

Type: String20

Properties: Searchable, PickList, IsNonEmpty.

Restrictions: Valid values:

- CheckedIn: The physical record is in storage at its home location. This value denotes when the physical record is stored at its "home location", the location where it is permanently stored. This data is changed if the record is loaned out or given to third parties.
- CheckedOut: The physical record has been transferred to a third party and is NOT in storage at its home location. This value denotes when a physical record has been transferred or loaned out to a third party and is not in storage at its "home location", the location where it is permanently stored. This data is changed if the record is loaned out or given to third parties. In this case, the metadata "sys:prm:CurrentLocation", "sys:prm:Custodian" and "sys:prm:ReturnDue" should also be updated.

Use: Folder, Document.

Description: This value represents the status of the physical record based in its current location or place of storage. It is set or edited if the record is loaned out or transferred to a third party who is storing it outside its home location.
This value can be edited throughout the entire entity life cycle. It is set by the custodian of the physical record, who must be given the ability to edit the entity's metadata.
The date and time of the most recent change are stored in the "sys:prm:StatusChange" metadata. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:prm:StatusChange«**

Type: DateTime

Properties: Searchable, ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

 

Use: Folder, Document.
Description: This value represents the date and time of the most recent change of status of the physical record; other changes are visible in the entity's audit trail. This value is determined automatically by the server when the value of the "sys:prm:Status" is edited and cannot be edited by the user. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:prm:HomeLocation«**

Type: String100
Properties: Searchable, IsNonEmpty.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Folder, Document.
Description: This value represents a description of the home location of the physical record. The exact "home location" where the records are permanently stored is entered in here (address, room, shelf, file cabinet, etc.) This value can be edited throughout the entire entity life cycle. It is set by the custodian of the physical record, who must be given the ability to edit the entity's metadata. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:prm:CurrentLocation«**

Type: String100
Properties: Searchable, IsNonEmpty.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Folder, Document.
Description: This value represents a description of the current location of the physical record if this location is different than the home location of the records or if the records are currently being borrowed by a third party.
The exact external location where the records are currently stored is entered here (address, room, shelf, file cabinet, etc.)
In this case it is also advisable to change the value of the "sys:prm:Status" attribute to "CheckedOut". This value can be edited throughout the entire entity life cycle.
It is set by the custodian of the physical record, who must be given the ability to edit the entity's metadata. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:prm:Custodian«**

Type: String100

Properties: Searchable.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Folder, Document.

Description: This value represents the ID of the current custodian of the physical record. If the records are stored at their home location (the "sys:prm:Status" attribute has the "CheckeIn" value), this will usually be the custodian of the physical records. If they are stored at an external location (the "sys:prm:Status" attribute has the "CheckedOut" value), this is the person who has been entrusted with the records for a limited period of time. This value can be edited throughout the entire entity life cycle.
It is set by the custodian of the physical record, who must be given the ability to edit the entity's metadata. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:prm:ReturnDue«**

Type: Date

Properties: Searchable.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Folder, Document.

Description: This value represents the date and time by which the physical record should be returned to its home location. The temporary custodian of the records listed in the "sys:prm:Custodian" attribute is responsible for returning the records. This value can be edited throughout the entire entity life cycle. It is set by the custodian of the physical record, who must be given the ability to edit the entity's metadata. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

### 3.2.8 Attributes of transferred records

When transferring the stored records from other electronic archives certain attributes of the transferred records need to be saved in the so-called system attributes in order to ensure the continuity of the life cycle of the stored records. Some system attributes of third-party systems are entered in IMiS®/ARChive Server system attributes.
This is not possible for some attributes, or standards do not allow it.

 

For this purpose, certain attributes are predefined on the server. They store these metadata values. During the process of "Importing" or "Transferring" records to IMiS®/ARChive Server, the server automatically assigns values for these attributes on the basis of information it has received from the client performing the operation.
The right to perform the "Import" operations is determined by the "ImportExport" role.
The right to perform the "Transfer" operations is determined by the "Reviews" role.
Except for their effect on the audit trail, transferred record attributes do not affect the business logic of the server in its operations with entities.

**»sys:trf:AuditLog«**

Type: StringMax
Properties: ReadOnly.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Class, Folder, Document.
Description: This value represents the audit trail of the entity from the previous system. Although this information is important for the continuity and auditing of the entity life cycle, this data is not required. Specifically, the third party system is not required to add this information to the metadata scheme of the exported entity when exporting. If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:trf:SystemId«**

Type: String100
Properties: ReadOnly.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Class, Folder, Document.
Description: This value represents a system identifier from the entity's previous system. Although this information is important for the continuity and auditing of the entity life cycle, this data is not required. Specifically, the third party system is not required to add this information to the metadata scheme of the exported entity when exporting.
This data can be used for the traceability of the instance of the entity from the previous ISDM to the instance of the entity in this ISDM.

 

If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:trf:ClassificationCode«**

Type: String100

Properties: ReadOnly, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: This value represents the classification code of the entity in the previous system. Although this information is important for the continuity and auditing of the entity life cycle, this data is not required. Specifically, the third party system is not required to add this information to the metadata scheme of the exported entity when exporting. This data can be used for the traceability of the instance of the entity from the previous ISDM to the instance of the entity in this ISDM.

If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:trf:Imported«**

Type: DateTime

Properties: ReadOnly, Searchable, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: This value represents the date and time the entity was imported if the “Import” function was used. Because the actual time of the creation of an entity (transfer from the previous ISDM) is recorded in the system attribute “sys:Created”, if the “Import” or “Transfer” function is used, the new ISDM must enter the date and time of the creation of the instance of the entity in the new ISDM.

This value is automatically set by the ISDM. It is not provided by the client.

If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM.

 

This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:trf:Evidence«**

Type: StringMax

Properties: ReadOnly.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

Description: This value represents an evidentiary record of the authenticity of the entity from the previous ISDM if the "Import" function has been used.

If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM.

This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:trf:MoveReason«**

Type: String200

Properties: Searchable, ReadOnly, Multivalue.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document

Description: These values represent reasons for each move (re-classification) of the entity in the previous ISDM (if the "Import" function was used). This value can be used in conjunction with the "sys:trf:MoveAgent", "sys:trf:MoveDateTime" and "sys:trf:MoveClassificationCode" attributes to provide more complete information about the movement of an entity in the classification hierarchy of the previous ISDM. If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:trf:MoveAgent«**

Type: String200

Properties: Searchable, ReadOnly, Multivalue.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Class, Folder, Document.

 

Description: These values represent directory entities (users) who have performed movements (re-classification) of an entity in the previous ISDM (if the "Import" function was used).
This value can be used in conjunction with the "sys:trf:MoveReason", "sys:trf:MoveDateTime" and "sys:trf:MoveClassificationCode" attributes to provide more complete information about the movement of an entity in the classification hierarchy of the previous ISDM. If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM.
This attribute does not affect the business logic of the server in operations with entities;
it is only a container of information.

**»sys:trf:MoveDateTime«**

Type: DateTime
Properties: Searchable, ReadOnly, Multivalue.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Class, Folder, Document.
Description: These values represent the date and time when an entity was moved (re-classified) in the previous ISDM (if the "Import" function was used). This value can be used in conjunction with the "sys:trf:MoveAgent", "sys:trf:MoveReason" and "sys:trf:MoveClassificationCode" attributes to provide more complete information about the movement of an entity in the classification hierarchy of the previous ISDM. If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:trf:MoveClassificationCode«**

Type: String200
Properties: Searchable, ReadOnly, Multivalue.
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Class, Folder, Document.
Description: These values represent the classification code of the entity when the entity was moved (re-classified) in the previous ISDM (if the "Import" function was used).
This value can be used in conjunction with the "sys:trf:MoveAgent", "sys:trf:MoveReason" and "sys:trf:MoveDateTime" attributes to provide more complete information about the movement of an entity in the classification hierarchy of the previous ISDM.

 

If the entity has been exported from an ISDM, this data is exported to the metadata scheme and can be re-imported to the attributes of the transferred entities by the third party ISDM. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

### 3.2.9 Attributes of retention policies and disposition holds

For retention policies, predetermined attributes and templates are on the IMiS®/ARChive Server. They are intended for saving metadata of retention policies.
Attributes are divided into the following groups:

- Attributes that affect operations of retention policies.
- Attributes that do not affect operations and are only containers of information.

Attributes and their role are described in the following section:

**»sys:ret:pol:Action«**

Type: Uint32

Properties: Public, Required, Searchable, Picklist, IsNonEmpty [ReadOnly in the case of Copies of retention policies]

Restrictions: Valid values:

- 1, alias »Dispose«: the default retention policy action is entity disposition;
- 2, alias »Permanent«: the default retention policy action is permanent entity retention;
- 3, alias »Transfer«: the default retention policy is a transfer of entities to another archive system and their disposition when successful transfer is confirmed;
- 4, alias »Review«: the default retention policy is to leave the entity for the next review process.

Use: Retention policies, Copies of retention policy

Description: A mandatory attribute that represents a default retention policy action used in the review process. Operations for individual actions are as follows:

- The »Dispose« action: if the user who performs the action has a right to delete an entity, the entity is irreversibly deleted as well as its metadata except for those that the deleted entity contains. The operation is irreversible which means the entity and its metadata cannot be reverted.
- The »Permanent« action: if the user who performs the action has editing rights on the entity, the entity is permanent. This entity can no longer be edited and creation of contained entities is not possible any more.

 

- The »Transfer« action: if the user who performs the action has entity deletion rights and has confirmed a successful entity transfer in the review process, the entity is irreversibly deleted. If the reference is recorded in the review process, the reference to the transferred entity is logged in »sys:ret:pol:Reference« in the deleted entity.
- The »Review« action: the entity is saved for the next review process.

All these actions are logged in the entity's audit trail together with a reason that is stated in the review process with a comment (if there is one).

**»sys:ret:pol:DetailedDescription«**

Type: StringMax
Properties: Public, [ReadOnly in the case of Copies of retention policies]
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Retention policies, Copies of retention policies
Description: An optional detailed description of retention period that is subject to change in the life cycle of an entity. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:ret:pol:Reason«**

Type: String200
Properties: Public, Required, IsNonEmpty [ReadOnly in the case of Copies of retention policies]
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Retention policies, Copies of retention policies
Description: A mandatory attribute. Its value is the default comment for an action in the review process. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:ret:pol:Reference«**

Type: String200
Properties: Public, ReadOnly, Searchable
Restrictions: There are no restrictions except those pertaining to the type or properties.
Use: Copies of retention policies.

 

Description: A mandatory attribute that is selected by a server in the review process and cannot be changed. It contains a reference to a retention policy that is included in the preparation process. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:ret:pol:Trigger«**

Type: String200

Properties: Public, Required, [ReadOnly in the case of Copies of retention policies]

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Retention policies, Copies of retention policies

Description: A mandatory attribute that must contain a valid condition for metadata search *(see chapter 3.5.2 Search syntax rules)*, otherwise the preparation process that contains a retention policy with an invalid condition aborts with an error.

**»sys:ret:hold:Reason«**

Type: String200

Properties: Public, Required, ReadOnly

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Hold of review peocesses

Description: A mandatory attribute that contains the reason for a hold of the review process. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

### 3.2.10 Attributes of review processes

For retention policies, predetermined attributes and templates for saving metadata on the review process are on the IMiS®/ARChive Server. As with retention policies and disposal hold *(see chapter 3.3.9 Retention periods)*, attributes are divided into those that affect only the implementation of review processes and those that are only containers of information. Attributes and their roles are described in detail below:

 

**»sys:ret:rev:Action«**

Type: UInt32

Properties: Searchable, Picklist, IsNonEmpty.

Restrictions: Valid values:

- 1, alias »Reviewing«: the value represents the action of review in the review process and it does not affect the server;
- 2, alias »Complete«: the value represents the action of completion in the review process on the server;
- 3, alias »Discard«: the value represents the action of cancellation in the review process on the server.

Use: Review process

Description: An optional attribute that is key for completing or discarding the review process on the server. The attribute use is as follows:

- When creating the review process, the attribute has no value.
- When the review process is in review, the attribute can change to »Reviewing«, but not necessarily. The review process is marked that it is in the process of reviewing.
- When the review is completed, the value changes to »Complete« or »Discard«.

In any case, the request goes to the action implementation queue.

In the case of »Complete« values, the review process will be implemented, whereas in the case of »Discard values«, the process will be discarded. In both cases, the status of review changes to »Closed«. Changing of the review process is disabled.

**»sys:ret:rev:Comments«**

Type: StringMax

Properties: /

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Review process

Description: An optional attribute that is used for entering comments, explanations and other information that is in any way connected to the review process. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:ret:rev:Lists«**

Type: File

Properties: Multivalue, ReadOnly, AppendOnly, IncludeInAIP, Searchable

Restrictions: There are no restrictions except those pertaining to the type or properties.
An entity can contain up to 65536 values.

Use: Review process.

Description: This attribute represents a container of contents. In the review process, the latter are XML documents that represent the result of the preparation of the review process. XML documents are prepared and recorded by the server and cannot be changed by the user. The attribute belongs to a group of indexed attributes.
What is special about the »File« attribute is that it represents the Full Text Index.
If the system for ensuring authenticity and inalterability during documentation retention is on, the fingerprint values from this container become part of the Archival Information Package (AIP).

**»sys:ret:rev:Members«**

Type: String200

Properties: Multivalue, Required, Searchable, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Review process

Description: This attribute is mandatory and represents the committee members that are present in the review process. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

**»sys:ret:rev:Message«**

Type: String200

Properties: Public, ReadOnly

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Review process

Description: This attribute is not mandatory. It is intended for the server to write a short error description if an error has occurred in the creation or implementation of the review process.
If the creation or implementation of the review process was successful, this is recorded in the value of the attribute. This attribute does not affect the business logic of the server in operations with entities; it is only a container of information.

 

**»sys:ret:rev:Query«**

Type: String200

Properties: ReadOnly, Searchable

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Review process

Description: The value of this attribute is not mandatory, but it must be present when the preview process is created and the value of the »sys:ret:rev:Schedules« attribute is not present. It is also not allowed for both attributes »sys:ret:rev:Query« and »sys:ret:rev:Schedules« to have set values.

The value must contain a valid condition for metadata search *(see chapter 3.5.2 Search syntax rules)*, otherwise the preparation of the review process that contains retention policies with an invalid condition aborts with an error.

**»sys:ret:rev:Schedules«**

Type: String200

Properties: Multivalue, ReadOnly, Searchable, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Review process

Description: The value of this attribute is not mandatory, but it must be present when the review process is created and the value of the »sys:ret:rev:Query« attribute is not present. It is not allowed for both attributes »sys:ret:rev:Query« and »sys:ret:rev:Schedules« to have set values.

The value must contain a valid condition for metadata search *(see chapter 3.5.2 Search syntax rules)*, otherwise the preparation of the review process that contains retention policies with an invalid condition aborts with an error.

**»sys:ret:rev:Scope«**

Type: String200

Properties: ReadOnly, Searchable, IsNonEmpty.

Restrictions: There are no restrictions except those pertaining to the type or properties.

Use: Review process

Description: The value of this attribute is not mandatory. It represents a classification code of the entity under which the review process will be implemented. If the value is not present, the preparation of the review process is implemented in the entire archive.

 

**»sys:ret:rev:State«**

Type: Uint32

Properties: Public, Required, ReadOnly, Searchable, Picklist, IsNonEmpty.

Restrictions: Valid values:

- 0, alias »Unknown«: the value of the attribute represents an invalid state of the review process;
- 1, alias »Created«: the value of the attribute is set by the server when the user creates a new review process;
- 2, alias »Preparing«: the value of the attribute is set by the server in the process of content preparation for the review process;
- 3, alias »InReview«: the value of the attribute is set by the server when the content has been successfully created for the review process;
- 4, alias »Completing«: the value of the attribute is set by the server at the beginning of review process implementation;
- 5, alias »Completed«: the value of the attribute is set by the server when the review process has been successfully completed;
- 6, alias »Discarded«: the value of the attribute is set by the server when the review process has been successfully discarded;
- 7, alias »Failed«: the value of the attribute is set by the server if a fatal error has occurred during implementation or discarding.

Use: Review process

Description: The value of this attribute represents the state of review process.

The »Preparing« value indicates that the server is currently creating content for the review process.

The »Completing« value indicates that the server is implementing the review process.

The »InReview« value indicates that the server has successfully implemented the content for the review process and that the user can review it, set actions, etc.

Changing the review process (for example, changing actions, entry of comments or reasons, etc.) is allowed only if the value of the attribute is set to »Created« or »InReview«. In other cases changing the review process is not allowed.

In the case of the »Failed« value, a message is recorded in the »sys:ret:rev:Message« attribute explaining why the implementation and creation of the review process was unsuccessful.

 

## 3.3 Entity

An entity is a free-standing container of data and content that forms a logical unit of information. An entity is uniquely identified using these three different identifiers:

- An internal or serial identifier (required, automatically generated when an entity is created).
- A classification code (required, generated automatically or manually when an entity is created).
- An external identifier (one or more, not required, externally generated, can be edited throughout the life cycle).

Each entity is defined by its:

- Type
- Classification code
- System attributes (name, description, life-cycle data, status, author, etc.)
- Entity-specific properties.

An entity is an abstract construct that saves the data and content of the object it saves (a folder, a physical document, a group of documents).
The different types of entities are specialized for the types of objects they describe or save. For this reason, they also have - besides system attributes - properties adapted to their objects.
At the same time, an entity is a container of access rights which determine which user actions are allowed or prohibited with the objects it saves.
Each entity has its own life cycle, which is recorded in its audit trail.

### 3.3.1 Types

The following types of entities have been defined in IMiS®/ARChive Server:

- Class
- Folder and Sub-folders
- Document.

 

Class

Classes are intended for sorting records on the basis of their content or the business activities of an organization. They are the basic building blocks of the classification scheme.
Classes can merge folders or documents by:

- Type of documents contained (for example, all invoices are located in one class, all purchase orders in another).
- Owner of the documents contained (for example, the personnel department keeps its documents in one class, the sales department keeps its documents in another, etc.)

The number of possible classes and the content of these classes are not prescribed by IMiS®/ARChive Server. The classification scheme manager can configure classes as they wish, in line with the needs of the organization using the archive server.

Folder

A folder represents a group of entities (subfolders, documents) that form a rounded-out whole in terms of content. It represents the "case file" on a given subject (a question, topic, task, project, etc.), complete with all properties and content.
A folder is the basic unit of merging, record keeping, sorting and archiving documents.

Document

The document represents a container for the properties and content it saves.
One document can save multiple contents (for example, text, images, video).
Though it is usually contained in folders and child folders, it can also appear as a free-standing document within a class. It represents the basic archive record unit for saving content.

### 3.3.2 Hierarchy

Every electronic archive is organized as a tree that starts with multiple roots (classes on the first level) and branches which represent individual entities.
The end classes contain the content entities (folders and subfolders, documents).
IMiS®/ARChive Server does not limit the depth of the classification scheme.

When defining a classification scheme, the following rules should be taken into consideration:

- One or more classes must be defined on the first level of the classification scheme. Adding folders or documents to the root of the tree must be disenabled.

 

- Every entity capable of containing other entities can only contain one type of subordinate entity.
- A class can contain sub-classes.
- The end class can contain folders or documents.
- A folder may contain subfolders.
- End folders can only contain documents.
- A document cannot contain child entities.

The hierarchies of entities in the classification scheme can also define access rights for all child entities *(see chapter 3.3.5 Access)*, except when access rights are defined in greater detail in the child entities themselves *(see chapter 3.3.5.2.5 Explicit Permission or Prohibition)*.
At the same time, the hierarchy defines the security class (when it is not explicitly defined; *(see chapter 3.3.5.1 Confidentiality (Security Class)* and whether an entity's status is required (the "sys:Status" attribute).

Status is defined for:

- Classes
- Folders and subfolders
- Documents filed into classes.

If a status value is not explicitly defined, it is inherited from the parent entity. This is also valid for classes on the first level, which receive the inherited value "Opened" if a value has not been explicitly defined.

### 3.3.3 Components

Entities consist of different components, including:

- A system attribute scheme.
- A specialized attribute scheme for the specific type of entity.
- A physical record management attribute scheme.
- A transferred record attribute scheme.
- A store with archived content and metadata about the content.
- Access right components.
- Authentication element components.
- Retention policies.

 

The structure of most components is determined by a template which serves as the basis for entity creation (a scheme of specialized attributes). Other components are systemic in nature and are always present because they are critical for the existence and life cycle of entities (a system attribute scheme, etc.) An entity is also the container of its access rights, audit trail and authenticity proof elements.

### 3.3.4 Templates

Templates prescribe a metadata scheme - required and allowed attributes.
At the same time, they are the basis for creating special templates for entities of the same type, which then inherit the attribute scheme.
Each template contains embedded and previously defined system attributes.
These attributes are required for the correct operations of IMiS®/ARChive Server and cannot be changed. Their names are derived from the fields to which they pertain: “int:”, “sys:”, “trf:”, and so forth.
All other attributes can be added to or deleted from templates by the administrator, until the first entity has been created. After that point, possibilities for changing templates become limited. These limitations are listed below.

When adding attributes to the template, an appropriately authorized user may specify whether:

- The attribute's value within the framework of the entity is publically accessible (Public).
- The attribute can have multiple values (Multivalue).
- The attribute is optional or mandatory (Required).
- The value of the attribute can be changed after it has been stored to the server (ReadOnly).
- The value of the attribute is inherited from the parent entity (Inherited).
  If the parent entity's values are not explicitly set, the values will be inherited from the first parent entity that has explicitly set values.
- Users can edit an already saved attribute or only append values to the attribute without deleting existing values (Append Only).
- The value of the attribute should be included in the archival information package (AIP) as part of document authenticity protection (IncludeInAIP).

 

Templates can be deleted only if there are no entities which would be rendered invalid by the deletion.
Other entity types (class, folder, document) must be assigned a template and must be built in accordance with the rules defined by the template's metadata scheme.
Templates can be newly created by the administrator, or they can be derived from other templates on the condition that the entity types are the same.

*Example: The "Invoice" template is used to create the "Incoming invoice" and "Outgoing invoice" templates. This way it is not necessary to define common attributes for each separate template. Additionally, all future changes to the "Invoice" template will be automatically taken into account in the templates derived from it, "Incoming invoices" and "Outgoing invoices".*

### 3.3.5 Access

Access to entities and entity management functions is of key importance, as it provides the basic principles of information security:

- Integrity: access is used to ensure that information is not changed.
- Confidentiality: access is used to ensure that information is accessible only to authorized persons.
- Availability: access is used to ensure the availability of information to authorized persons.

Access right checking is a two-step process:

- Checking the security class.
- Checking access to entities (the Access Control List - ACL).

Security class is a requirement for viewing whether an entity exists and for performing actions with the entity. The administrator can define the maximum security class level for which each user or each group is authorized.
The user must have the same or a higher security class level as the entities they can view.
Security class is specifically defined for entities, or inherited from parent entities.
If a user's security class level enables them to determine the existence of entities, their ability to access these entities is checked.

IMiS®/ARChive Server uses Access Control Lists (ACL) to check user access rights.
The system checks whether a user attempting to perform an operation on the server has the right to do so.

 

If the user does not have the right to perform the operation, the operation is not performed and the server returns an error. The illustration below shows how access works.

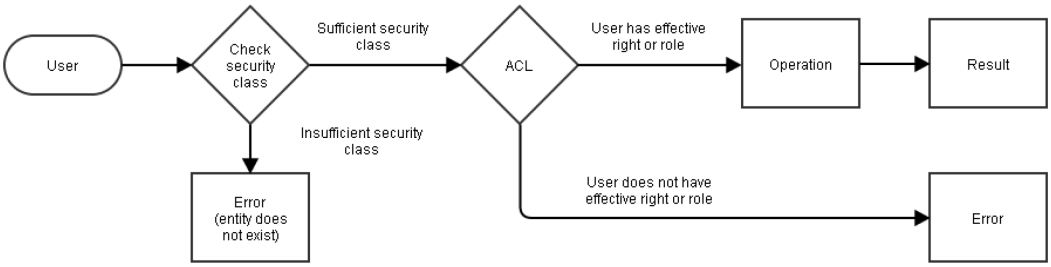


Image 3: How ACLs work

#### 3.3.5.1 Confidentiality (Security class)

The degree of confidentiality or security class provides an additional level of security for access to archived records. It is used to prevent the disclosure of documents to unauthorized persons. The administrator can define security class on the level of documents and can specify which security classes each user (or group) is allowed to view. In the basic settings, all document types are accessible to all users (degree of confidentiality is 0). As in the case of access rights, the security class is inherited from the top down within the hierarchy.
If an entity in the hierarchy has an explicitly defined security class, the explicit value overrides the inherited one.

The following rules exist for defining confidentiality:

- The highest security class a user can set for an entity is equal to the calculated effective security class.
- For a given entity, a user can only set a security class lower or equal to the security class of the parent entity. If a security class has not been defined for the parent entity, the user can enter a security class for this entity that is equal to or lower than user's effective security class.
- When the security class is raised, it is also raised for all child entities with the same security class as the entity whose security class has been raised.
- When the security class is lowered, it is also lowered for all child entities with a greater security class than the entity whose security class is greater than that set by the user.

 

*Example 1: Let's refer to the hierarchy in the image Access – Effective Rights. Let's assume that a security class has not been explicitly defined for any of the entities (all entities have a degree of confidentiality of 0). Let's set a security class of 2 (Restricted) for the class. Because of inheritance, the security class for all child entities is then set to 2.*

*Example 2: We would like to raise the security class of the folders in the previous example to 3 (Confidential). Because the parent class has a security class of 2, we cannot raise the security class.*

*Example 3: We would like to lower the security class of the folders in the previous example to 1 (Unrestricted). The security class of both documents is lowered to 1 due to inheritance.*

*Example 4: We would like to raise the security class of the class to 3 (Confidential). The security class of the folder and the child documents is not changed because the newly set security class is only taken into account for those entities, which have the same security class as the entity whose security class has been changed.*

*Example 5: Let's raise the security class of the class to 2 (Restricted). The security class of the documents is also raised to 2. We then lower the security class of the class to 1 (Unrestricted). The security class is also lowered for the folder and the documents because the security class is changed for all child entities with a higher security class than the security class we just set.*

As noted above, the highest security class a user is able to set depends on their calculated effective security class. As in the case of calculating effective rights, an effective security class is defined for each user, depending on the groups to which the user belongs.

The effective security class is calculated in the following way:

- If a security class has been explicitly set for a user, this value takes precedent.
- If a security class has not been explicitly set for a user, their effective security class is calculated on the basis of the groups to which the user belongs.
  The effective security class is the highest security class of all groups to which the user belongs (regardless of their position in the entity hierarchy).

 

*Example 1: A security class of 1 (Unclassified) has been explicitly set for a user, although this user is in two groups, one with a security class of 2 (Restricted) and 3 (Confidential). This user's effective security class is 1, because the explicitly set security class takes precedent over the security classes of the two groups. This user can access entities accessible to all users (security class of 0) and entities whose security class is set to 1 (Restricted).*

*Example 2: Let's take the user from the previous example and delete the security class we explicitly set for them. This user's effective security class is now 3 (Confidential), as the highest security class of the groups to which the user belongs now takes precedent. This user therefore has access to entities with a security class of 3 or lower.*

#### 3.3.5.2 Access Control List (ACL)

A user with the right to define access rights for an entity (Change permissions) can assign an Access Control List to a user or a user group. Access rights are divided into two groups:

- Access rights to an entity.
- Special access rights to an entity.

An authorized user can explicitly set Allow or Deny for each group of rights. These values are valid for an individual right within the group, and can also be set for a limited period of time. Explicit permissions and inherited rights are used to determine effective rights, that is, the rights of a user to perform requested operations on the server.

 

#### 3.3.5.2.1 Access rights for an entity

The following access rights are used for classes, folders and documents:

- Read: The right to view an entity.
  The user must have this right to open an entity in read-only mode.
- Write: The right to edit in an entity.
  Besides Write rights, to edit an entity, the user must also have Read rights (Read-write mode).
- Move: The right to move an entity in the classification scheme.
  The user is allowed to move an entity and its child entities in the classification scheme.
- Delete: The right to delete an entity.
  The user is allowed to delete an entity and its components.
- Create entities: The right to create new child entities or sub-entities.
  This right allows users to create new entities inside the entity.
- Change permissions: The right to edit the Access Control List (ACL).
  The user is allowed to change access rights for entities and metadata.
  Changing access rights is only possible if the entity is open for editing.
- Change security class: The right to change the security class.
  Users with this right can set the initial security class of an entity before the entity has been saved, or they can change the security class of existing entities. If a user does not have this right, they cannot set the initial security class (it is implicitly inherited from the parent entity) or change it at a later time.
- Change status: The right to change the status. Users granted this right can change the status of the entity. The default status value is inherited from the parent.
  The two available explicit values of the status are "Opened" and "Closed".
  When an entity's status is "Closed", this will also set all its child entities to "Closed".
  When a previously closed entity is set to "Opened", the status only changes for this specific entity, while its child entities preserve their initial status.
- Change retention: The right to change the retention policies.
  Users with rights can change the validity of entity retention policies. Default values are inherited from the parent entities.

 

The table below shows operations with entities and the access rights required to perform them.

| Operation | Access right | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Read | Write | Delete | Move | Change-permissions | Create-entities | Change security class | Change status | Change retention |
| Opening entities in Read mode | ✓ | | | | | | | | |
| Editing entities | ✓ | ✓ | | | | | | | |
| Deleting entities | ✓ | | ✓ | | | | | | |
| Editing the ACL | ✓ | ✓ | | | ✓ | | | | |
| Re-classification (source entity) | ✓ | | | ✓ | | | | | |
| Re-classification (target entity) | | | | | | ✓ | | | |
| Creating entities (subordinate entity) | | | | | | ✓ | | | |
| Changing the security class | ✓ | | | | | | ✓ | | |
| Changing status | ✓ | | | | | | | ✓ | |
| Changing the retention policies | | | | | | | | | ✓ |

Table 4: Operations and rights

#### 3.3.5.2.2 Access rights to metadata

The following access rights are used for entity metadata:

- Read: The right to view metadata.
- Write: The right to edit metadata values.
- Delete: The right to delete metadata.
- Create: The right to assign metadata values to previously empty metadata.

 

The server also uses access rights to metadata to limit access to and editing of metadata that are not public. So a user with the Read right for a given entity is prevented from reading individual metadata by denying the Read right for metadata. If access rights have not been defined for metadata, the metadata takes them from the access rights to the entity to which the metadata belongs.
Denying the Delete right for metadata does not prevent the deletion of an entity and its metadata.
If a user has the right to delete an entity but does not have the right to delete its metadata, the right to delete the entity takes precedent over the prohibition on deleting the entity's metadata.

#### 3.3.5.2.3 Exceptions

The following metadata are exempted when setting permissions and prohibitions:

- Public metadata.
- Metadata marked as "read only".
- Required metadata.
- Special system metadata.

Public metadata does not contain classified information, which is why it is an exception when checking access rights. Settings cannot be used to deny users the right to read metadata of this kind, and permissions cannot be set without the Read access right. The user can also read the metadata if they do not have Read rights for the metadata's entity if IMiS®/ARChive Server global security settings enable this.

 

The table below shows restrictions on users' rights to view public metadata. These limitations differ based on the user's Read rights for the entity and global security settings.

| Restrictions on viewing public metadata | Read access right for an entity | Global security setting |
|---|---|---|
| The user will not see any information, and cannot affirm the existence of an entity (even if they have the appropriate security class). | | |
| The user sees public metadata. The user can confirm the existence of the entity, but cannot view its non-public metadata. | | ✓ |
| If the user has read rights for the entity, they also have the right to read public metadata regardless of global security settings. | ✓ | N/A |

Table 5: Restrictions on viewing public metadata

Metadata marked as Read Only cannot be assigned Write and Read rights. Create are explicitly set for required metadata, and Delete rights cannot be set for this type of metadata.
Besides the exceptions listed above, the following system metadata are treated separately: sys:ExternalIds, sys:Opened, sys:Closed, sys:Creator. No permissions or prohibitions can be set for these metadata.

#### 3.3.5.2.4 Roles

A role is a package of rights that enables a user to perform certain operations on the server. The following roles are predefined:

- AuditLogQuery: This role enables a user to obtain the audit trail.
- ImportExport: This role enables a user to import and export records.
- Deletion: This role enables the viewing of entities that have been slated for deletion.
- Reports: This role enables:
  - The display of system reports on imports and exports.
  - The display of system reports on access, folders, documents, contents and retention periods.
  - Printing of data on classes, folders and documents.
  - Printing of classes (and folders of the classification scheme).

 

- Review: this role enables a display of reviews in a review process.
- AdminCodeListaRead: this role enables a review of codelists.
- AdminCodeListUpdate: this role enables adding, deleting or changing the codelists.
- AdminCountersRead: this role enables a counter settings review.
- AdminCounterUpdate: this role enables adding, deleting or changing counter settings.
- AdminDirectoryEntitiesRead: this role enables a review of directory entities.
- AdminDirectoryEntityUpdate: this role enables adding, deleting or changing the directory entity.
- AdminDirectoryGroupRead: this role enables a preview of directory group members.
- AdminDirectoryGroupUpdate: this role enables adding or deleting directory group members.
- AdminAttributesRead: this role enables a review of attributes.
- AdminAttributeUpdate: this role enables adding, deleting and changing an attribute.
- AdminTemplatesRead: this role enables reading of entity templates.
- AdminTemplateUpdate: this role enables adding, deleting or changing entity templates.
- AdminStorageProfilesRead: this role enables reading of server profiles.
- AdminStorageProfileUpdate: the role enables adding, deleting or changing server profiles.
- AdminStorageVolumesRead: this role enables reading of server volumes.
- AdminStorageVolumeUpdate: this role enables adding, deleting or changing server volumes.
- AdminACLRead: this role enables reading of access rights.
- AdminACLUpdate: this role enables adding, deleting or changing access rights.
- AdminArchiveSettingsRead: this role enables reading of general archive settings.
- AdminArchiveSettingsUpdate: this role enables adding, deleting or changing general archive settings.
- AdminAuditLogSettingsRead: this role enables reading of audit trail settings.
- AdminAuditLogSettingsUpdate: this role enables adding, deleting or changing audit trail settings.
- AdminRetentionRead: this role enables reading of retention policies.
- AdminRetentionUpdate: this role enables adding, deleting or changing retention policies.

 

#### 3.3.5.2.5 Explicit permission or prohibition

A user's effective rights can be managed by setting explicit permissions and prohibitions.
Every change made to permission or prohibition is logged in the audit trail.
Permissions and prohibitions have the following properties:

- For each group of access rights, permissions or prohibitions can be set for each user or group.
- Permissions or prohibitions can be set for a limited period of time.

Time limits on permissions and prohibitions can be divided into:

- Permissions or prohibitions without a time limit.
- Permissions or prohibitions with a time limit starting at a specified point.
- Permissions or prohibitions with a time limit ending at a specified point.
- Permissions or prohibitions with a time limit starting and ending at a specified point.

Permissions or prohibitions without a time limit

This is the standard way to use permissions and prohibitions. Permissions and prohibitions of this kind are always valid, regardless of when they are used to calculate a user's effective rights.

Permissions or prohibitions with a time limit starting at a specified point

A starting point is defined for permissions or prohibitions, but not an end point.
These permissions and prohibitions are used to calculate effective rights if the current date is later than or on the date of the start of the validity of the permission or prohibition.

Permissions or prohibitions with a time limit ending at a specified point

An end point is defined for permissions or prohibitions, but not a starting point.
These permissions and prohibitions are used to calculate effective rights if the current date is before or on the date of the end of the validity of the permission or prohibition.

Permissions or prohibitions with a time limit starting and ending at a specified point

A start and end point are defined for permissions or prohibitions.
These permissions and prohibitions are used to calculate effective rights if the current date is later than or on the date of the start of the validity of the permission and before or on the end of the validity of the permission or prohibition.

 

Explicit permissions or prohibitions are automatically inherited for all subordinate entities in a given hierarchy.

#### 3.3.5.2.6 Effective rights

Effective rights are a package of inherited and explicit permissions and or prohibitions that tells us whether a user has the right to perform an operation in IMiS®/ARChive Server.

The order of priority in which permissions or prohibitions are taken into account when calculating effective rights is as follows:

1. Explicit deny rights.
2. Explicit allow rights.
3. Inherited deny rights.
4. Inherited allow rights.

Calculations of effective rights are not affected by the number of groups in which a user is included, but they are affected by the hierarchy of groups in the Access Control List.
The hierarchy shown in the illustration below presents the calculation of effective rights.

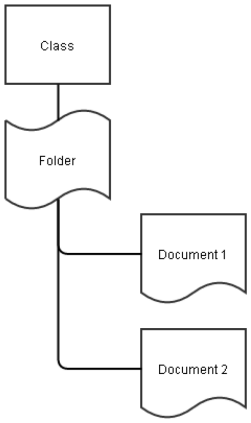


Image 4: Hierarchy with a class, folder and 2 documents

*Example 1: We would like to set read rights for the entire hierarchy shown in the illustration below for a particular user. In the Access Control List for the class, let's set an explicit Read right for this user. Inheritance in the hierarchy enables this user to have the effective right to read in all subordinate entities in the class. So the user in our example can open the class, the folder and both documents in Read-only mode.*

 

*Example 2: Let's also give the user in example 1 the right to edit the folder and both documents in the folder. To edit entities, the user will need both Read and Write rights. The user has already inherited Read rights from the class. That is why we only need to set an explicit Write right for this user in the Access Control List of the folder. This will give the user effective Read and Write rights for the folder and for both documents within the folder.*

*Example 3: Because of the actions performed in examples 1 and 2, the user has rights to view all entities in the hierarchy and to edit the folder and both documents within the folder. We would now like to deny the user the ability to edit Document 1. So we set an explicit Write deny right in the Access Control List. Now the user cannot edit this document, as the deny right has priority over the inherited allow rights when calculating the user's effective access rights. The user still has the right to edit the folder and Document 2, but cannot edit Document 1.*

*Example 4: In example 3 we prevented the user from editing Document 1. Now let's add an explicit Write allow right for the user in the Access Control List. This will not affect the calculation of effective rights, as the explicit Write deny right will take priority over the explicit allow right. As a result, the user will have the same effective rights as in the previous example.*

*Example 5: In the access control list for the folder, let's give the user the explicit allow right to edit the Access Control List (the Change permissions right). The user receives the effective right to edit the ACL in the folder and in Document 2. The user still cannot edit the ACL in Document 1 because the explicit Write deny right prevents this.*

*Example 6: We would like to revoke the right of the user from example 5 to edit the Access Control List (ACL). At the same time we would like to prevent them from editing certain metadata in the folder and the two documents because this metadata is not of a public nature. In the ACL of the folder, we set an explicit deny right for editing metadata. This prevents the user from editing the metadata, even though they have the right to edit the folder and Document 2.*

*Example 7: Let's revoke the user's explicit Read right for the Access Control List (ACL) for the class. This will prevent the user from opening and editing all entities in the hierarchy.*

*Example 8: Let's remove all of the user's explicit allow and deny rights from the Access Control List (ACL) for all entities in the hierarchy. In the ACL for the class we add explicit Read, Write and Delete rights for the entity and an explicit Delete deny right for all metadata in the hierarchy. This will enable the user to edit all entities and metadata in the hierarchy, but will prevent them from deleting metadata. The user cannot delete metadata, but they can delete an entire entity including the entity's metadata because the right to delete the entity has priority over the prohibition on deleting the entity's metadata.*

 

*Example 9: Let's set a starting point on the explicit metadata Delete deny rights we set in example 8: this deny right will take effect on 1. 4. 2015; there is no end point.*
*The user will lose the effective right to delete metadata on 1.4.2015; before this date, the user can normally delete metadata.*

*If we set an end point for the restriction with the same date (with no starting point), the user will not have the effective right to delete metadata before 1.4.2015; on and after 1.4.2015 the user will have this right, as the deny right will no longer be taken into account when calculating effective rights, and the right to delete metadata is taken from the entity.*

*Example 10: Let's clear the Access Control List (ACL) for all entities in the hierarchy, removing all explicit allow and deny rights. Then let's set the following ACL values for the user:*

- *In the ACL for the class, let's set an explicit Read allow right for the entity from 1.4.2015 to 15.4.2015.*
- *In the ACL for the folder, let's set explicit Write and Create sub-entities allow rights for a limited period of time, from 6.4.2015 to 12.4.2015.*
- *Now let's set an explicit Write deny right in the ACL of Document 1, and let's define a time period for this right, 8.4.2015 to 12.4.2015.*

In line with the values we have entered in the Access Control List (ACL), the user now has the following rights for specified periods of time:

- From 1.4.2015 to 15.4.2015, the user has the right to open all entities in the hierarchy.
- From 6.4.2015 to 12.1.2015, the user has the right to edit the folder and Document 2 and the right to create child or sub-entities in the folder.
- The user is allowed to edit Document 1 from 6.4.2015 to 8.4.2015, as the explicit Write deny right for this document takes effect on 8.4.2015.
  The explicit Write deny right has priority when calculating effective rights, and this overrides the effective Write allow right.
- The user does not have the right to open and edit entities in the hierarchy between 1.4.2015 and 15.4.2015. The user still has the right to read public metadata if the global security settings allow this.

 

### 3.3.6 Identifiers

IMiS®/ARChive Server uses three methods for identifying individual entities. With each identification method, each identifier specifies one and only one entity; an unknown identifier can also specify no documents in the hierarchy. An identifier can never specify more than one entity.

#### 3.3.6.1 Internal identifiers

An entity's internal identifier is an automatically generated string 24 to 32 bytes in length (the length depends on the requirements of the client).
The content of the string depends on the identifier of the archive, which should be unique for each archive. This ensures that the unique identifiers of individual entities are also globally unique.
IMiS®/ARChive Server uses AES-256 encryption when generating strings, and this ensures that the probability of randomly guessing the correct identifier is negligibly small.
In an archive with 100,000,000 entities, the probability that a randomly generated identifier will be correct is 1 : 1,593 x $10^{50}$ for identifiers 24 bytes long 1 : 8,636 x $10^{70}$ for identifiers 32 bytes long.

For saving identifiers when the binary method of saving is not possible, the server enables three methods for encoding internal identifiers:

- Hexidecimal encoding, where the identifier is represented by a string of characters from 0 to 9 and from a to f. This string is 48 characters long for 24-byte identifiers and 64 characters long for 32-byte identifiers.
- Base64 encoding, where the identifier is represented by a string of characters taken from the lower- and upper-case letters of the English alphabet (52 characters), numbers (0-9), hyphens (-) and underscores (_). This string is 32 characters long for 24-byte identifiers and 43 characters long for 32-byte identifiers.
- Base85 encoding, where the identifier is represented by a string of characters taken from the array of base64 characters and 21 additional characters: !#$%&()*+;<=>?@^`{|}~. An internal identifier of this kind is 30 characters long for 24-byte identifiers and 40 characters long for 32-byte identifiers.

The encoding method is determined by client requests based on the requirements of the database model where the identifiers will be stored.

 

#### 3.3.6.2 External identifiers

External identifiers are custom character strings up to 100 characters in length which are generated independently of the IMiS®/ARChive Server environment. The server enables the association of an entity with any number of external identifiers on the condition that no entity in a given archive is associated with the same identifier.
An entity is not required to be associated with an external identifier.

#### 3.3.6.3 Classification code

For faster work with entities in the classification scheme and for enhanced transparency, entities (classes, folders, documents) are labeled in the classification scheme.
IMiS®/ARChive Server automatically assigns classification codes to entities in the classification scheme based on the position of classes and folders in the hierarchy. Classification codes are uniformly defined and are assigned at installation or at a later time by managing the classification scheme.
If the position of a class is changed in the classification scheme, all entities filed under this class are given new identifiers that show their new position in the hierarchy. The new classification code immediately becomes valid for all entities filed in the class.
When the server cannot assign a classification code to a newly created entity because of an inadequate configuration, or because the configuration has been purposefully set up this way, the user must define a classification code when the new entity is created.
This classification code must also be uniquely defined within the parent entity; if it is not, the server will deny the request for the creation of the new entity and will return an error.
The request will be also denied and an error will be returned if the user has not defined a classification code.
When moving records within an archive (re-classification), the classification codes of all moved entities must be changed to correspond to their new positions within the hierarchy. In this case, classification codes cannot be manually added and the server will deny a request to move an entity if a new classification code cannot be automatically assigned to any of the moved entities.

 

#### 3.3.6.3.1 Fully Qualified Classification Code – FQCC

*Example: **C=01^C=02^F=2014-01^D=0001***

The canonical form of the fully qualified classification code is used mostly for communications between IMiS®/ARChive Server and clients; users rarely see classification codes in this form. The canonical form of the fully qualified classification code consists of multiple components, each of which represents an entity's own/partial classification code from the hierarchy to which the entity belongs.
The components are always separated by the "^" sign. Each individual component consists of two parts. The first part is a single-character code for the entity type: C for class, F for folder and D for document. The second component is the actual value of the classification code; its two parts are separated by the = sign.

#### 3.3.6.3.2 Public Classification Code – PCC

*Example: **01.02-2014-01/0001** (equal to the canonical form in the previous example)*

The public classification code is the form that users see as the full classification code. It is made up of the same number of components as the canonical form, but each component only contains the actual classification code value. The components include characters the product administrator has defined as separators in classifications codes, with a different character for each entity type.
In this example, components that represent classes are preceded by a period (.), components that represent files are preceded by a minus (-), and the final component, which represents a document, is preceded by a backslash (/).

### 3.3.7 Life cycle

There are two different types of entity life cycles. The first describes a change of status, and the second is only the life cycle of the instance within a user session.
The information presented below is valid for the class, folder and document entity types unless a specific entity type is listed.

 

#### 3.3.7.1 Entity status

When an entity is created, it receives Opened status (for a detailed description *see chapter 3.3.2 Hierarchy*). The user can create and assign custom statuses by changing attribute values. Once an entity has been saved for the first time, the attribute values can no longer be changed. The user can change the status by performing an action to change the status.
In the Opened status, the user can perform operations on entities that enable entity attributes to be changed and child entities to be added (for example, entering new documents in a folder). The user can change the status to Closed only when they are fairly sure that the entity will no longer be changed and new entities will not be created under the entity. The user cannot edit the content of closed entities. In a closed entity and its subordinate entities *(see chapter 3.3.7.1.2 "Closed" Status)* the user can launch the authentication process *(see chapter 3.6.1 Conditions)*. This makes it possible to create and maintain authenticity proof elements, which would otherwise become invalid in storage if changes were later made to the entity.
An entity can be deleted at any phase of the life cycle. Deletion before the disposition process is exceptional and should not be a part of regular processes at an organization. It should only be used if an entity has been incorrectly entered, as it is intended for error correction.

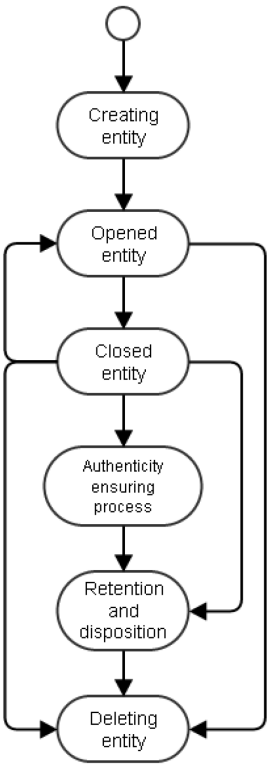


Image 5: Entity life cycle

 

#### 3.3.7.1.1 "Opened" status

Opened status enables the entity to be opened in Write mode and makes it possible for new child entities to be added to the entity (for example, creating new documents in a folder).
A user opens a closed entity by changing the status of only the entity they would like to work with; the status of all child entities remains closed.

A closed entity can only be opened if:

- The entity the user would like to open is closed.
- And has opened child entities.

#### 3.3.7.1.2 "Closed" Status

An entity with Closed status can only be opened in Read-Only mode. It is not possible to edit the attributes or to add child entities. If authentication settings have been entered in IMiS®/ARChive Server, the server will include entities of this kind in the list of entities suitable for generating authenticity proof elements. Closing an opened entity causes the closing of all open child entities with a status. If the closed entities (and the entities they contain) are included in an authentication process, an Archival Information Package (AIP) is generated for all edited entities and the authentication process is launched (*see chapter 3.6.1 Conditions*).

### 3.3.7.2 Entity instance

An instance is a representation of an entity in IMiS®/ARChive Server's working memory.
When opening an entity, the system must first check whether the entity is already in the cache. It is always in the cache when it is being used in another session.
If the entity is not in the cache, it is loaded from the database; otherwise the already existing instance, which is shared by all sessions, is used. This model enables the efficient use of server resources and faster responses to requests to open entities, as the use of already loaded instances is incomparably faster than transferring them from the database.
If a user has Read access, a new copy of the entity will be created in the cache, and this copy will not be accessible to any other session. If another session already has its own copy of the entity opened (the entity is opened in Write mode in another session), access is denied.
Access is also denied if the entity is marked as permanent.
The other sessions will access the instance from which the copy for writing was made;
this copy remains active for all the other sessions and cannot be edited. At the same time, only one copy "for writing" is allowed.

 

Consequently, no change made to the entity will be visible in other sessions until the process for saving the entity opened for writing is performed. This process checks the consistency of the entered metadata, saves changes to the database and replaces the instance in the cache. After saving, all new operations for opening the entity return the changed instance.
Sessions where the entity was opened before saving will not see the new version in the existing instance (the model operates on the principle of an unchangeable status at the moment of opening).

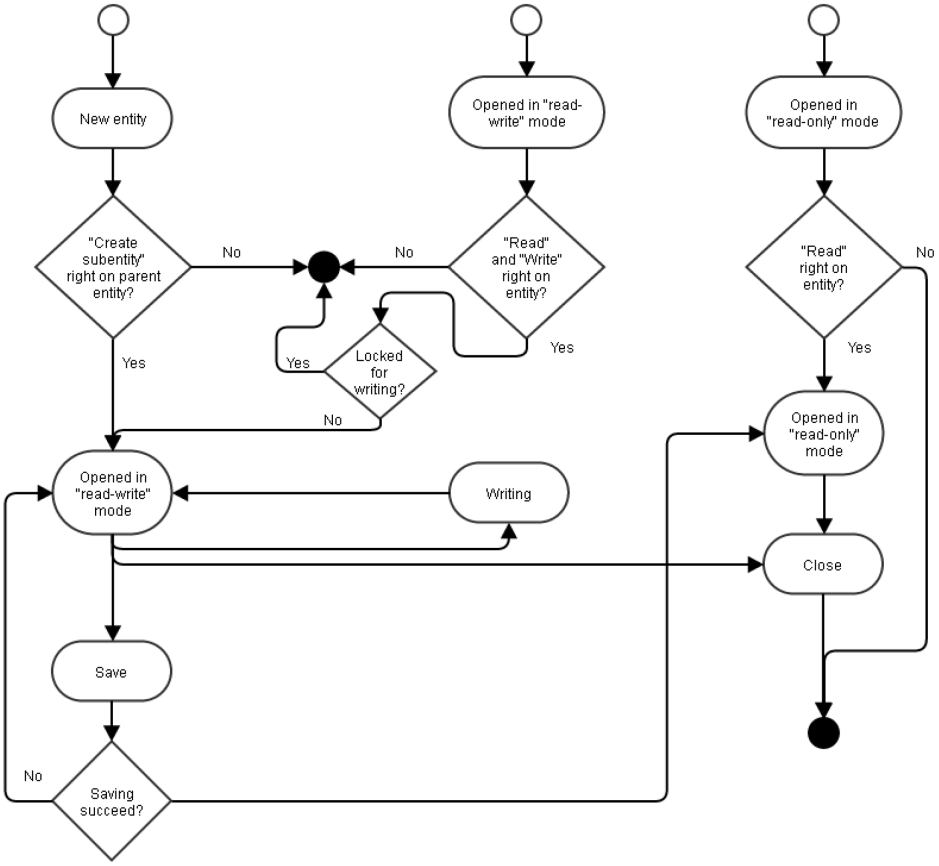


Image 6: Entity life cycle in the working memory of the server

### 3.3.7.3 New entity

An unsaved instance of an entity is created as a result of a request from a client which must contain:

- The valid unique identifier of the parent entity.
- The valid unique identifier of the template, which cannot be a system or internal template.

 

It also checks if the following conditions have been met:

- The right to access the parent entity as determined by the security class.
- The user has the right to create new child or sub-entities in the parent entity.
- The parent entity cannot have status “Closed”.
- The parent entity must not be marked for permanent retention *(see chapter 3.7 Review process)*.
- The unique template identifier must be included in the list of allowed entities in this part of the classification scheme.

If any of these conditions has not been met, IMiS®/ARChive Server will return an error.
It will create a new instance of the entity in RAM, but this instance is not recorded in the database in this phase.
The instance is accessible only in the session of the user who created the object. If a request is made for the new entity from another session, the instance will not be visible.
The entity type determines the template used, and the template cannot be changed.
A classification code for the entity has not been defined in this phase. The entity is physically saved on the server once the method for saving has been called up.
After a successful save, the entity becomes accessible to other sessions.

#### 3.3.7.4 Opening an entity in Read-Write or in Read-Only mode

In IMiS®/ARChive Server, a request to open an entity is sent by a client. The client determines whether the entity will be opened in Read-Only or RO mode or in Read-Write or RW mode.
The server opens the requested instance of the entity (*see chapter 3.3.7.2 Entity Instance*) and checks access rights.
If the security class of the entity allows it to be opened and if the user has the required access rights, the server will return:

- A reference to the opened instance of the entity.
- An identifier of the entity’s parent entity.
- The template with which the entity was created.
- Different kids of system metadata on the entity (date created, last edited, effective rights of the user for the entity, etc.).
- List of templates which can be used to create child entities.

 

All later requests must refer to the reference given.
The entity-opening event leads to the creation of a record of the event in the audit trail.

#### 3.3.7.5 Reading entity content

A request to read data from an entity is a demand that IMiS®/ARChive Server read the components of the entity. The entity must be opened in either Read-Only mode or Read-Write mode. The request must contain (at least):

- A reference to the entity instance.
- The extent of the data being read.

The extent of the data being read contains instructions that tell the server what its reply to the request should contain.
The request can contain a demand for:

- Metadata (all, public, listed).
- Access rights (records of access rights for the entity, records of access rights for the attributes, or both).
- Proof of the authenticity and integrity of the entity (AIP and ERS).

In its reply to the client, the server returns all the values of the requested attributes, except for those for which the user does not have access rights.
If the user does not have access rights, IMiS®/ARChive Server does not return an error, but it also does not return the metadata for which the user does not have rights.
Reading entity content is an event that does not lead to a record in the audit trail.

#### 3.3.7.6 Editing entity content

A request to log changes to the components of the instance of an entity is a demand that IMiS®/ARChive Server save changes to the metadata in the request to its instance of the entity in memory. A client can send multiple requests for editing an entity that is open for writing. Changes are not saved to the database until the server receives a request to save the entity.

 

A request contains:

- A reference to the entity instance (required).
- A list of special system attributes and their values.
- A list of attributes and their values (all attributes of the entity except for special system attributes).
- A list of any changed access rights (added, changed, deleted).

If a user changes an attribute that contains multiple values, all the values must be logged - both those that have been changed and those that remain the same. When logging these values the system also partially checks the validity of the attribute values *(see chapter 3.2.2 Parameters)*.
On the server, multiple consecutive requests for editing in an entity that is open for writing can be sent. In its reply to the request for changes, the server either gives an affirmative reply or returns an error stating that the changes could not be made. The entity is physically saved on the server once the method for saving has been called up. After a successful save, the entity becomes accessible to other sessions.

The event in which the entity is changed does not lead to a record in the audit trail, any changes are logged during saving, because changing an entity only changes the instance of the entity in working memory, and these changes are not permanent.

#### 3.3.7.7 Saving

A save request is a demand for IMiS®/ARChive Server to save an entity. An entity open for writing is physically recorded in the database, and a write-proof copy is loaded in the cache *(see chapter 3.3.7.2 Entity instance)*.
The request must contain (at least) a reference to the entity instance.

When logging these values the system also partially checks the validity of the attribute values *(see chapter 3.2.2 Parameters)*. The reason for the two-phase verification is that the validity of all attributes cannot be checked when saving. A typical example is checking for the presence of required attributes; this is not possible in an individual call to check an instance.
In the process of saving, effective entity retention periods are checked, as well as whether the entity represents a class, folder or document under class.
If the entity has no effective retention period, saving is not successful.

 

After an instance of an entity has been saved, the entity remains open in Read-Only mode, and the client receives the entity's unique internal identifier, which the client or application can save to the data collections of third party applications to directly call up the entity at a later time. Following successful saving, changes to the entity are visible in every instance opened following the saving. Instances opened before saving remain unchanged throughout their entire life cycle.

Saving creates a record of the following events in the audit trail:

- "Change of attribute value": if a change has been made to one or more system or special attributes.
- "Change of access control list": if a change has been made to one or more records of access rights.
- "Change of physical record attribute": if a change has been made to one or more attributes for physical records management.
- "Entity saved".

#### 3.3.7.8 Closing

A request for closing an instance of an entity is a demand for IMiS®/ARChive Server to reduce the number of references to the entity instance by 1; if the counter of reference goes to 0, it deletes the instance in the cache.
A request for closing is also made for all instances of an entity in a session when the session is closed, provided an explicit request for closing an entity instance(s) has not been made.
The request must contain (at least):

- A reference to the entity instance.

Following the closing, the reference to the entity instance becomes invalid.
If an unsaved instance of an entity that was open in Write mode is closed, any changes will be permanently lost. Closing does not lead to a record in the audit trail.

#### 3.3.7.9 Move

The movement of an entity is triggered with a move request. Moving a document means classifying the entity in a different part of the classification scheme; moving can therefore also be called re-classification.

 

The request must contain (at least):

- The unique identifier of the entity the user would like to move.
- The unique identifier under which the user would like to classify the entity in the previous sentence.
- A reason for the move.

Once an entity has been moved, IMiS®/ARChive Server checks the following conditions:

- The right to access the parent entity as determined by the security class.
- The right to edit the entity being moved (adding system attribute values).
- The right to move the entity being moved.
- The right to add new child entities to the entity to which the moved entity is being moved (new location).
- The target entity where the entity being moved is being filed, must allow for the creation of child entities using the template with which the entity being moved was created.
- The target entity where the entity is being moved must not be closed.

If any of these conditions has not been met, IMiS®/ARChive Server will deny the move.
All child entities are moved along with the selected entity.
All the above listed conditions are also checked for the child entities.
Classification codes for the moved entities are recalculated in accordance with the rules valid for the new location *(see chapter 3.4.1 Classification codes)* and have no connection to the old classification codes. The user who performed the move is required to enter a reason for the move.
The move event is recorded in the audit trail of the moved entity, and a record of the move event is also created in all child entities moved together with the main entity.
The volume of data in the report in the audit trail differs depending on the entity being moved and its child entities.

#### 3.3.7.10 Changing the security class

Security class is changed when a client sends a request to change security class.
The request must contain (at least):

- The unique identifier of the entity.
- The new security class.
- The reason for the change.

 

Once the security class of an entity has been changed, IMiS®/ARChive Server checks the following conditions:

- The right to access the entity as determined by the security class.
- The right of the user to change the security class in the entity.
- The right to use a security class equivalent to the user's; the new security class must be equal to or less than the user's.
- The entity whose security class is being changed cannot be closed.

The server first checks if the operation to change the security class can be performed.
The new security class may not be greater than the security class inherited from the entity's parent entity. If the security class is raised, it is changed for the entity and for any of its child entities with the same security class; in so much as the child entities have a lower security class, they are unchanged. If the security class is lowered, the security class is changed to the new value for all entities for which a security class has been explicitly specified.

When changing the security class, the server automatically fills in the attributes of the changes of the security class in the entity for which a new security class has been defined:

- "sys:SecurityClassChangeReason": The reason for changing the security class is given by the user; this data is required when changing the security class.
- "sys:SecurityClassChangeAgent": The user who performed the change of the security class.
- "sys:SecurityClassChangeDateTime": The date and time of the change of the security class.
- "sys:SecurityClassChangeFrom": The value of the effective security class BEFORE the change; it can be explicit or inherited.
- "sys:SecurityClassChangeTo": The value of the effective security class AFTER the change; it can be explicit or inherited.

Every change of the security class that is either a direct change or a change as the result of an entity's child status is logged in the audit trail of the entity as a security class change event; the record contains the old value, the new value and the reason for the change.

 

#### 3.3.7.11 Deleting

The deletion of an entity is triggered with a delete request. The request must contain (at least):

- The unique identifier of the entity.
- If the entity's sys:Description attribute does not have a value, one can be entered upon deletion, as the description is required information when deleting an entity.
- The reason for the deletion.

Before an entity can be deleted, IMiS®/ARChive Server checks the following conditions:

- The right of the user to delete the entity.
- The presence of all required metadata: the sys:Description system attribute, which becomes required in the event of deletion, is also checked.
- The sys:Significance system attribute is also checked; if this value is set to 1 ("Vital") or 2 ("Permanent"), deletion of the entity will not be possible.

The method of deletion can be set in the server so that:

- The integrity of the entity is maintained, that is, none of the entity's components are deleted *(the Moreq2 9.3.1 Specification)*.
- All metadata and content is deleted except for those required to maintain the consistency of the deleted entity *(the Moreq2 9.3.1 Specification)*.

In both cases, the entity is moved to the Deletion system class (classification code C=sys^C=Trash^C=Deleted), where all deletions are collected.
As a result, the entity is "hidden" from all users and is accessible only to users with the Deletion role.

Every deleted entity, regardless of the settings for the extent of the deletion, maintains the following system attributes:

- sys:Title: The title of the entity.
- sys:Description: The description of the entity (this data is otherwise not required, but becomes required in the event of a deletion)
  When the user enters a description of the entity prior to deletion, the entity is opened in editing mode, the attribute value is updated and the entity is saved. All these changes are recorded by the audit log.

 

The server also automatically adds the following system attributes to the deleted entity:

- int:Template: The original template with which the entity was created (hidden from user).
- int:ParentId: The internal identifier of the parent entity where the entity was located (hidden from user).
- sys:del:Reason: The reason for deleting the entity; this data is entered by the user and is required when deleting an entity.
- sys:del:Agent: The user who performed the deletion.
- sys:del:DateTime: The date and time of the deletion.
- sys:del:ClassificationCode: The classification code of the entity prior to deletion.
- sys:del:Reference: Reference to the transferred entity. The attribute is empty if an *entitiy is deleted. It is used only for transferred entities in the review process* (*see chapter 3.7 Review process*).

*Warning: Once performed, a deletion is irreversible.*

If the description of an entity is provided at the time of deletion, records of following events will be logged in the audit trail:

- The opening of the entity in Write mode.
- The sys:Description system attribute was changed.
- The entity was saved.

The deletion event is logged in the audit trail of the deleted entity.

#### 3.3.7.12 Changing status

A client can request a change of status.
The request must contain (at least):

- The unique identifier of the entity.
- The new value of the status.
- The reason for the change (not required).

Before the status of an entity is changed, IMiS®/ARChive Server checks the following preconditions:

- The right of the user to change the status of the entity and any child entities.
- The parent entities of the entity whose status is being changed must be open.

The change of status can be broken down into three actions:

 

- Opening a closed entity.
- Closing an open entity.
- Setting the inherited value in the open entity.

Opening a closed entity: Opening a closed entity causes only the entity in question to be opened; its child entities remain closed. The new status is entered in the sys:Status system attribute, and the date and time of the opening are entered in the sys:Opened system attribute; the sys:Closed value is deleted. The opening event is logged in the audit trail of the open entity.

Closing an opened entity: Closing an opened entity causes the entity in question and all its open child entities to be closed. The attribute sys:Closed is set for all closed entities using the current date and time, and the new status value is recorded in the sys:Status attribute (only for the entity in question; child entities inherit this value). A status change event is logged in the audit trail of every closed entity.

Setting inherited values: If an open entity has an explicitly set status value, this action will give it the inherited value from its parent entity. Because the content value of the status was not changed for the entity in question, this action is not logged in the audit trail.

### 3.3.8 Audit trail

The audit trail is an undeletable chronological record of access, queries and changes made in IMiS®/ARChive Server. The minimum information the audit trail contains is the user, the time and the action performed on any document, folder or class in the classification scheme.
It is also a documented record about certain performed procedures.
The audit trail is completely inalterable throughout its entire life cycle.
In this regard, it is protected from both authorized and unauthorized interventions.

It provides a log of changes and an overview of procedures performed.
It is intended for the performance of activities on the archived objects.
The data in the audit trail are saved together with the archived records on the server.
For each user session, an audit session with a start and end is also logged.
All events contain a reference to this audit session.

#### 3.3.8.1 Sessions

 

When a session is opened in IMiS®/ARChive Server an audit session is also opened.
When the audit session is opened, the following information is recorded in the database:

- The user account of the user who logged in.
- The hostname of the computer from which the session was launched.
- The date and time of the start of the session.
- The internal network address (the internal IP address of the network interface from which the session was launched; it is provided by the client in the authentication data).
- The public network address (the network address of the client as seen by the server).
- The date and time of the end of the session.
- The reason for ending the session:
  - Session was not closed (0)
  - Normal session end (1)
  - Unexpected termination of the session by the client (2)
  - Time-out (3)
  - Improper authorization (4)
  - Improper request (5).

Once this data has been logged, the server assigns the session a unique identifier which remains the same for the duration of the session and which is used to log every event detected and launched in the framework of the session; it becomes linking data for data about the session and data about all events in the session.

#### 3.3.8.2 Events

Every access to the server is logged as an event in the audit trail.
In every record of an event in the audit trail, the following parameters (minimum) are logged:

- Unique audit session identifier.
- Event type.
- Date and time of the event.

 

The audit trail allows the following non-required parameters to also be saved; they are provided by the client when an action is performed:

- Reason for action: can have a "printf" format message in the parameter (*http://en.wikipedia.org/wiki/Printf_format_string*).
- Reason for action parameters: if the reason contains parameters, the content of the parameters in the print out of the action are merged with the reason for the action in a uniform message.

The following events are automatically logged in the audit trail:

New entity [1]

This event is logged in the audit trail whenever a user creates a new entity (*see chapter 3.3.7.3 New entity*).

Because the entity requires a unique identifier, which at this point is not available (it is assigned during saving), the event is physically recorded in the database only when it is saved with the time of the saving, at the same time as the entity saving event.

The reason for using this logging model for this type of event is also connected to the fact that before saving, the entity does not actually exist, and it only begins to exist on the server once it has first been saved.

The following are recorded in the audit trail report:

- Unique identifier of the entity.
- Reason/message from the user when an entity is created (not required).
- Reason/report parameters (not required, required if a reason/report with parameters is present).

Opening an entity in Read-Only mode [2]

This event is logged in the audit trail each time a user opens an entity in Read mode (*see chapter 3.3.7.4 Opening an entity in Read-Write or in Read-Only mode*).

 

The following are recorded in the audit trail report:

- Unique identifier of the entity.
- Reason/report from user for opening the entity (not required).
- Reason/report parameters (not required, required if a reason/report with parameters is present).

*Opening an entity in Read and Write mode [3]*

This event is logged in the audit trail each time a user opens an entity in Write mode (*see chapter 3.3.7.4 Opening an entity in Read-Write or in Read-Only mode*).

The following are recorded in the audit trail report:

- Unique identifier of the entity.
- Reason/report from user for opening the entity (not required).
- Reason/report parameters (not required, required if a reason/report with parameters is present).

Entity saved [4]

This event is logged in the audit trail when the entity is saved (*see chapter 3.3.7.7 Saving*).

The following are recorded in the audit trail report:

- Unique identifier of the entity.
- Reason/report from user for saving the entity (not required).
- Reason/report parameters (not required, required if a reason/report with parameters is present).

Entity moved [5]

This event is logged in the audit trail when the entity is moved (*see chapter 3.3.7.9 Move*).
For each movement of an entity (movement of a branch of entities can lead to multiple movements) an event is logged in the audit trail.
The following are logged in the audit trail report:

- Entity status.
- Full old classification code.
- Full new classification code.
- Values of all contained attributes.
- Reason/report from user for moving the entity (not required).

 

Deleting an entity [6]

An event is logged in the audit trail when an entity is deleted *(see chapter 3.3.7.11 Deleting)*. Required information about the reason for the deletion is entered in both the specific attribute for this purpose and in the audit trail.

The following are logged in the audit trail report:

- Unique identifier of the entity.
- Reason/report from user for deleting the entity (not required).
- Reason/report parameters (not required, required if a reason/report with parameters is present).

Audit trail query [7]

This event is logged in the audit trail every time a query is made for an entity in the audit trail.
The following is logged in the audit trail report:

- Search term created by the server on the basis of the user's query.

Change audit trail settings [8]

This event is logged in the audit trail when the server detects that audit trail settings have been changed during server off-time (by directly manipulating settings record).
Because changes to the settings of the audit trail can only be made when the service is not running, this check of the identity of the previous and new audit trail settings is performed when the audit trail starts up.
The audit trail report records the settings that were changed and all settings. Changed settings are marked with a *.

*Example: The following are logged in the audit log:*

- *Enabled state changed to on*.*
- *Global required audit log parameters settings changed. UserName: on* ComputerName: on, PrivateAddress: on, Reason: on.*
- *Events settings changed. AuditLog.Query: on*, Entity.Create: on*, Entity.OpenReadOnly: on*, Entity.OpenReadWrite: on*, Entity.Update: on*, Entity.Move: on*, Entity.Delete: on*, Entity.ACLChange: on*, Entity.PropertiesChange: on*, Entity.PhysicalRecordsManagementChange: on*, Entity.SecurityClassChange: on*, Entity.StatusChange: on*, Entity.Disposed: on*, Entity.Permanent: on*, Entity.Transferred: on*, Entity.Reviewed: on*, Content.OpenReadOnly: on*, Content.OpenReadWrite: on*, Content.Create: on*, Content.Delete: on*, Content.Update: on*, Content.MetadataChange: on*.*

 

The logging of this event prevents abuse, for example, an administrator with access to the server configuration could change the temporary server settings.

*Example: Turning off the audit trail for the duration of one query.*

Change attribute value [9]

An event is entered in the audit trail when an entity is saved, that is, when the value of one or more attributes is changed *(see chapter 3.3.7.6 Editing entity content* and *chapter 3.3.7.7 Saving)*. Physical record management attributes are listed in their own event and this event is not logged.

The following are logged in the audit log:

- Unique identifier of the entity.
- List of all attributes that have been changed.

Changing the Access Control List [10]

An event is entered in the audit trail when an entity is saved if the Access Control List (ACL) has been changed *(see chapter 3.3.7.6 Editing entity content* and *chapter 3.3.7.7 Saving)*.
The server checks and compares the existing Access Control List with the new one and looks for differences.

If a change has occurred, both lists are logged in the audit trail: the old one and the new one.
If a new entry was added to the list, only this entry is recorded, as there is no old value.
If a record is deleted from the access control list, only the old record is recorded,
as there is no new value.
The unique identifier of an attribute is logged whenever the Access Control List is changed for that attribute.

The following are logged in the audit log:

- Unique identifier of the directory entity - user or group.
- Old status of the record in the Access Control List.
- New status of the record in the Access Control List.
- Unique identifier of the attribute - if the record in the access control list pertains to an attribute.

 

Changing physical record management attributes [11]

An event is entered in the audit trail when an entity is saved *(see chapter 3.3.7.6 Editing entity content* and *chapter 3.3.7.7 Saving)*. An event is logged only if at least one physical record management attribute has been changed.

The following are logged in the audit log:

- Unique identifier of the entity.
- All changed and unchanged physical record management attributes and their values before and after the change.

Changing the security class [12]

This event is logged in the audit trail every time the security class is changed *(see chapter 3.3.5.1 Confidentiality (Security Class))*.

The following are logged in the audit log:

- Old security class.
- New security class.
- Notes.

Changing entity status [13]

This event is logged in the audit trail every time an entity's status is changed.

The following are logged in the audit log:

- New status value.
- Reason (not required).

Changing retention policies and disposition holds in the review process [14]

This event is recorded in the audit trail when effective retention policies and additional or removed entity holds are changed in review process.

The following titles are logged in the audit log:

- Added retention policies
- Removed retention policies
- Added holds
- Removed holds.

 

Opening content in Read mode [17]

This event is recorded in the audit trail when content is opened in Read mode

(for a detailed account of content *see chapter 3.2.5 System attributes - sys:Content*).

The following are logged in the audit log:

- Content identifier.
- Description of content (if one exists).
- Reason for the change (not required).

Opening content in Write mode [18]

This event is recorded in the audit trail when content is opened in Write mode.

The following are logged in the audit log:

- Content identifier.
- Description of content (if one exists).
- Reason for the change (not required).

Creating content [19]

This event is recorded in the audit trail when content is created.

The following are logged in the audit log:

- Content identifier.
- Description of content (if one exists).
- Reason for the change (not required).

Deleting content [20]

This event is recorded in the audit trail when content is deleted.

The following are logged in the audit log:

- Content identifier.
- Description of the content (if one was present at the time of deletion).
- Reason (not required).

Saving content [21]

This event is recorded in the audit trail when edited content is saved.

The following are logged in the audit log:

 

- Content identifier.
- Description of content (if one exists).
- Reason (not required).

Changing content metadata [22]

This event is recorded in the audit trail when content metadata is changed (description is changed).

The following are logged in the audit log:

- Content identifier.
- New description of the content (if one is present).
- Reason (not required).

Entity disposition [23]

This event is recorded in the audit trail when an entity is disposed in the review process.

The following are logged in the audit log:

- Reason for disposal.
- Comment (not required).

Permanent entity retention [24]

This event is recorded in the audit trail when the entity is checked as permanent in the review process.

The following are logged in the audit log:

- Reason for permanent retention.
- Comment (not required).

Entity delivery [25]

This event is recorded in the audit trail when the entity is delivered in the review process.

The following are logged in the audit log:

- Reason for delivery.
- Comment (not required).

Omitted for the next review [26]

 

This event is recorded in the audit trail when the entity is dropped in the review process.

The following are logged in the audit log:

- Reason for omission.
- Comment (not required).

#### 3.3.8.3 Right to view the audit trail

A user can perform audit trail queries if they have the right to view the audit trail. A user either has the right to view the entire audit trail or they do not. This right is obtained by assigning the AuditLogQuery role. Viewing the audit trail results in an XML file *(see chapter 3.3.8.5 Report format)*.

#### 3.3.8.4 Query

When making a query, a user can use the following search parameters to limit the results to events that are of particular interest to them:

- Event date range.
- IP address range (for example, from 192.168.1.1 to 192.168.1.99).
- List of IP addresses.
- List of user names.
- List of names of computers used to access the system.
- List of encoded unique entity identifiers.

If the search parameters are not selective enough and return too many results, the server returns an error. If this occurs, the user is advised to repeat the query with more precise or different search parameters. The easiest way to limit the range of results is to the shortest range of dates possible.
Search parameters can also be combined, but there are limits.
Auditlog query condition can always contain a date-time range condition.
One additional parameter can also be selected from the following list:

- Range of IP addresses.
- List of IP addresses.
- List of user names.
- List of names of computers through which the system was accessed.

Viewig the audit trail results in an XML file *(see chapter 3.3.8.5 Report format)*.

 

#### 3.3.8.5 Report format

The report created on the basis of data in the audit trail is crucial for reconstructing the events that led to a change of archived records at any point in the life cycle - from their creation to the viewing of the audit trail. In the absence of an audit trail, it is practically impossible for an authorized person to conduct an audit based on subjective evaluations of circumstances and events.

The results of the overview of the audit trail are provided to the client who performed the overview in the form of an XML file. The file is assembled in accordance with the XSD scheme standards included in IMiS®/ARChive Server. The XSD scheme is also available at *http://www.imis.si/imisarc/auditlog.xsd*.

*Example: The XML record contains the results of an audit trail query*

```
<?xmlversion="1.0" encoding="UTF-8"?>
<auditlog.query.resultset xsi:schemaLocation="http://www.imis.si/imisarc
http://www.imis.si/imisarc/auditlog.xsd" xmlns="http://www.imis.si/imisarc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <sessions>
    <!--Audit query sessions.-->
        <sessionid="3" closureReason="2" address="192.168.92.77"
        internal_address="192.168.92.77" dateTimeOpened="2014-04-23T14:51:38Z"
        dateTimeClosed="2014-04-23T14:54:00Z" username="jnovak" computerName="novak-pc"/>
        <sessionid="4" closureReason="2" address="192.168.92.23" internal_address="192.168.92.23"
        dateTimeOpened="2014-04-23T14:54:25Z" dateTimeClosed="2014-04-23T15:33:10Z"
        username="fkovac" computerName="kovac-pc"/>
        <sessionid="23" closureReason="0" address="192.168.92.77"
        internal_address="192.168.92.77" dateTimeOpened="2014-04-23T14:54:25Z"
        username="jnovak" computerName="novak-pc"/>
    <sessions>
    <events>
    <!--Audit query events.-->
    <!--Sort compare function is QuerySorter::CompareSESS_TS-->
        <eventseq="0" sessionId="23" type="2" dateTime="2014-04-24T10:01:47Z"
        classificationCode="C=01"
        context="0D=R}rSc^&amp;lt;N3ZQRdCGJqtf&amp;amp;z2iB`~aWgCI+MeIYx"/>
        <eventseq="1" sessionId="3" type="1" dateTime="2014-04-23T14:51:43Z"
        classificationCode="C=01"
        context="0D=R}rSc^&amp;lt;N3ZQRdCGJqtf&amp;amp;z2iB`~aWgCI+MeIYx"/>
        <eventseq="2" sessionId="3" type="4" dateTime="2014-04-23T14:51:43Z"
        classificationCode="C=01"
        context="0D=R}rSc^&amp;lt;N3ZQRdCGJqtf&amp;amp;z2iB`~aWgCI+MeIYx"/>
        <eventseq="3" sessionId="3" type="10" dateTime="2014-04-23T14:51:43Z"
        classificationCode="C=01" context="0D=R}rSc^&amp;lt;N3ZQRdCGJqtf&amp;amp;z2iB`~aWgCI+MeIYx"
        message="Values for the following properties have changed:sys:Creator, sys:Opened,
```

```
        sys:Owner, sys:Status, sys:Title"/>
        <eventseq="4" sessionId="4" type="2" dateTime="2014-04-23T14:54:26Z"
        classificationCode="C=01"
        context="0D=R}rSc^&lt;N3ZQRdCGJqtf&amp;z2iB`~aWgCI+MeIYx"/>
    </events>
</auditlog.query.resultset>
```

The XML files is made up of two sets:

- "Sessions"
- "Events".

#### 3.3.8.5.1 Information about sessions: "Sessions"

"Sessions" contain a list of all sessions within which any of the searched events were launched.

List of session data:

- Unique session identifier (ID).
- Reason for closing the session (closureReason); a list of possible reasons for closing a session in the XSD format is available at *http://www.imis.si/imisarc/auditlog.xsd*.
- Public IP address (address).
- Private IP address (internal_address).
- Name of computer (computerName).
- Date and time the session was opened (dateTimeOpened).
- Date and time the session was closed (dateTimeClosed).
- Unique user account identifier - user (username).

If the session has not been closed, the value of the reason for closing the session will be equal to 0 and the time of closing will be unspecified. This occurs when a session is still active or (theoretically speaking) the server was stopped in an uncontrolled manner during the session.

 

#### 3.3.8.5.2 Information about events: "Events"

"Events" contain a list of searched events and their data.

List of event data:

- Sequence of the event in the list (seq).
- Session identifier (sessionId): a reference to the session during which the event was launched.
- Event type (type).
- Time of the event (dateTime).
- Classification code of the relevant object (classificationCode).
- Encoded unique entity identifier (context).
- Message/reason for the action (message).

### 3.3.9 Retention periods

Retention periods represent a time frame in which the archive system must retain entities. When the time frame passes, a decision is made what will happen with retained entities in the review process.

Each retention period contains the following attributes alongside the mandatory attributes that are key for its operations in the review process:

- the »sys:ret:pol:Trigger« attribute represents a search string to determine a time frame of a retention period *(see chapter 3.5.2 Search syntax rules)*;
- the »sys:ret:pol:Action« attribute represents the default action for entities that will become the subject of review when the retention period will have passed.

Each entity in the classification scheme (except the document in the folder) needs at least one defined retention period. By establishing connections between entities and retention policies, effective retention periods are monitored and so is the preparation of the review process, though indirectly.

The disposition hold has a special role in the review process. It enables a disposal from the process of preparation of all entities that are connected to at least one hold.

In the process of implementation, a selected action is not implemented for such entities.

As with retention policies, holds are managed through connections that are described below.

#### 3.3.9.1 Managing disposition hold connections

 

Connect the disposition holds to the entities when you want to hold them from the review process. A disposition hold can be connected to all types of entities that can be also connected to retention polices. It applies to an entity to which it is connected and to all contained entities regardless of effective retention policies. It is valid until it is cancelled or until the connection is deleted.

*Example: For an example, take a folder with a contained folder »F=2015-00011^F=00001« that is in the decision of the review process. Due to an event (e.g., a lawsuit), the »F=2015-00011« folder is key documentation that must not be destroyed during the process. In this case, bind the disposition hold to the »F=2015-00011« folder. By doing so, we make sure that the folder with a contained folder is disposed from the review process. Also, the folder with a contained folder will not be a subject to the preparation of the review process until there is still an existing connection to the disposition hold.*

#### 3.3.9.2 Managing retention policies connections

The following parameters can be specified when connecting retention policies to the entities:

- Whether a policy is enabled or disabled.
- Types of entities the retention policies affect (class, folder, document classified directly under a class).

Effectiveness of retention policies of individual entities is monitored with these parameters. Effective retention policies are a set of explicit and inherited retention policies that tell us which policies effectively affect the review process of individual entities.
The order of importance of retention policy connections when calculating effectiveness is the following:

1. Explicit connections
2. Inherited connections.

Effective retention periods are calculated in the following way:

- Retention policy connections from the entire branch of entities (all parent entities) are reviewed. With hierarchy in mind, it is checked which retention policies affect the current type of entity for which they are calculated. Inherited retention periods are the result.

 

- Explicit connections to the retention policies that are joined with inherited retention policies are checked for the entity in question. Effective retention periods that apply to this entity are the result.

*Examples: The table below shows an example of a retention policy configuration.*

| Retention policy | Time frame |
|---|---|
| 5 years | 5 years since entity creation |
| 10 years | 10 years since entity creation |

Table 6: Example of a retention policy configuration

*We have the following classification tree:*

*C=3300*
*C=3300^C=3301*
*C=3300^C=3301^F=2015-00100*
*C=3300^C=3301^F=2015-00101*
*C=3300^C=3301^F=2015-00102*
*C=3300^C=3302*
*C=3300^C=3302^D=1000*
*C=3300^C=3302^D=1001*

*The main class is »C=3300« and it contains two classes, »C=3301« and »C=3302«. The contained class »C=3301« contains folders »F=2015-00100«, »F=2015-00101« and »F=2015-00102«. The contained class »C=3302« contains documents »D=1000« and »D=1001«.*

*Example 1: In the main class »C=3300«, the bound »5 years« retention policy is enabled for classes, folders and documents classified directly under classes. All entities in the tree are covered with this configuration as the retention policy is inherited to all types of entities in the tree.*

*Example 2: Change the configuration from Example 1 so the »5 years« retention policy is enabled for classes and folders. Such a configuration is invalid as the retention policy is inherited to all types of entities in the tree except documents under a class because they do not have explicit connections to the retention policies.*

*Example 3: Add an explicit connection for the »10 years« retention policy to the contained class »C=3302« and enable it for documents. At the same time, enable the »5 years« retention policy for classes and folders in the class »C=3300«.Such a configuration is appropriate as the »5 years« retention policy applies to all classes and folders, whereas the »10 years« retention policy is valid only for documents in the contained class »C=3302«.*

 

*Example 4: Connect the »10 years« retention policy to the class »C=3300« and enable it for classes and documents. Connect the »5 years« retention policy to the contained class »C=3301« and enable it for folders. Such a configuration determines that subjects will be selected according to the »5 years« retention policy, whereas classes and contained classes with documents will be selected according to the »10 years« retention policy.*

*Example 5: Connect the »10 years« policy to the class »C=3300« and enable it for classes and documents. Connect the »5 years« policy to the same class and enable it for folders. Such a configuration enables selection of all folders within the class according to the »5 years« retention policy, and class and document selection under them according to the »10 years« retention policy.*

*Example 6: Connect the »10 years« policy to the class»C=3300« and enable it for classes, folders and documents. Connect the »10 years« retention policy to the contained class »C=3301« and disable it for classes and folders. At the same time, connect the »5 years« retention policy and enable it for classes and folders.*
*With such configuration, you can select classes, folders and documents under classes with the »10 years« retention policy except the subclass »C=3301« which has a disabled »10 years« retention policy. However, it has an enabled »5 years« retention policy for classes and folders.*

*Example 7: Connect the »10 years« retention policy to the class »C=3300« and enable it for classes, folders and documents. Then connect the »5 years« retention policy to the contained class »C=3301« and enable it for folders. Configuration is valid as all types of entities have effective retention policy in the tree, however folders in the contained class »C=3301« are in conflict as both retention policies are inherited (the »10 years« policy is inherited from the main class »C=3300« and the »5 years« policy from the contained class »C=3301«). In the case of selection to one or both retention policies, folders will always be present due to the conflict.*

*Example 8: Connect the »10 years« retention policy to the class »C=3300« and enable it for classes, folders and documents. Connect the »10 years« policy to the folder »F=2015-00100« and enable it for folders. At the same time, connect the »5 years« retention policy to the same folder and enable it for folders. Follow the same procedure for folders »F=2015-00101« and »F=2015-00102«. With such configuration, you can select folders according to the »5 years« retention policy, and classes and documents under classes according to the »10 years« retention policy.*

*Example 9: Connect the »10 years« retention policy to the class »C=3300« and enable it for classes, folders and documents. Connect the »10 years« policy to the contained class »C=3302« and disable it for documents. The configuration is invalid as documents in the contained class do not have effective retention policies.*

 

*If you bind, for example, the »5 years« retention policy to both documents in the contained class and enable it for documents, then you can disable the »10 years« retention policy for documents in the contained class »C=3302« as all documents in the subclass have at least one effective retention policy*

*Example 10: The following configuration is in the entity tree: the enabled »10 years« policy is bound to the class »C=3300«. The »5 years« policy is bound to the contained class »C=3301« and it applies to folders. The »5 years« policy is also bound to the contained class »C=3302« and it applies to documents.*
*You want the »10 years« policy to apply to all entities in the tree of entities. The easiest way is to enable the »10 years« policy to the »C=3300« class for classes, folders and documents. This causes a conflict for folders and document, but it can be solved by removing the connections to policies on the subclasses »C=3301« in »C=3302«. At the same time, the policy configuration on the »C=3300« class covers the entire tree of entities with the »10 years« retention policy.*

## 3.4 Classification

Documentation classification is an irreplaceable tool even for traditional document management, that is, for paper documents.
The same is true for digital management and electronic archives. A documentation classification scheme must be established in advance for the needs of classification in IMiS®/ARChive Server.

The scheme enables the following:

- Structuring, breaking down and sorting documentation by content, authorizations, activities and business and professional functions.
- Setting documentation storage dates.
- Defining the archive records, forms of documentation and metadata structure added to folders and documents, including the management of records on physical records.

The server enables the user to set up a classification scheme with a practically limitless number of dimensions. It does not limit the possible number of class levels or the number of subclasses in individual classes. The number of class levels in different parts of the archive can vary.
The classification scheme is the basis for overseeing access to individual parts of the archive and is represented in the archive by a hierarchy of entities of the class type.

 

### 3.4.1 Classification codes

Every entity in the archive has its own full classification code, which is unique throughout the entire archive. It is assigned when an entity is created and cannot be changed unless an entity is moved in the archive (re-classification).

The full classification code of a class consists of the full classification code of the parent class and the code that unique only to this entity. A separator can optionally be added; the separator is uniformly used throughout the archive and is set by the archive administrator in the server settings (".", for example).

The full classification code of a folder consists of the full classification code of the class where the folder is classified. A unique code is added to the folder within this class. A separator can optionally separate the two components; the separator is set by the archive administrator and can be different than the one that separates class levels. It is set by the archive administrator in the server settings ("–", for example).
The full classification code of a document consists of the full classification code of the folder and class to which the document belongs. A unique identifier for the document within the parent entity is added to this. A separator can optionally appear before this code; it is set by the archive administrator and is intended for separating the components of a document in full classification codes. It is set by the archive administrator in the server settings ("/", for example).

IMiS®/ARChive Server automatically generates classification codes when new entities of all types are added *(see chapter 3.3.6.3 Classification codes)*.
If a classification code has not been automatically assigned to the entity (as per the settings of an individual entity for its child entities) the server will deny the request for creating a new entity. It does this if a classification code value has not been added to the request or if the submitted classification code is not unique to the new entity within the parent entity.

 

### 3.4.2 Classification scheme settings

A detailed and practically finite classification scheme must generally be set during the initial configuration of IMiS®/ARChive Server or at least prior to use.
The administrator adds all required classes to the hierarchy of the classification scheme.

Before adding classes, the following steps must be taken:

- Templates must be created for the new classes (*see chapter 3.3.4 Templates*).
- Settings for automatic classification code generation must be entered (as needed).

Before folders and documents are created, the following must be prepared:

- Templates for files in end classes (classes without subclasses).
- Templates for documents that will be created in the folders or directly in the classes.

Templates are used to define:

- Metadata required for individual entities.
- Metadata that can be entered.
- Metadata where multiple values can be set for the same entity.
- Other properties of the entity.

Each template can specify which templates can be used to create child entities in the entity created using the template. Templates that can be used to create child entities can also be directly specified in already existing archive entities. So classes in the last level of classes in the archive usually contain folders, and can exceptionally also contain free-standing documents. Every class can always contain only one type of entities, that is, only subclasses or only folders or only documents.

Precisely preparing a classification scheme prior to using the server is very important. IMiS®/ARChive Server does not allow templates to be replaced for entities that have already been created and saved. It also does not allow changes to templates that have already been used to create entities; following the start of server use, once folders and/or documents already exist on the server the following operations which affect the classification scheme are allowed:

 

- New classes can be added to any level in the archive at any time.
  New classes will be assigned classification codes automatically or manually, depending on the automatic classification code settings.
- A class can be deleted if the class contains no entities, that is, if it is completely empty.
- Adding templates that define the metadata of child entities. They can be added to a particular location in the hierarchy.
- Entities can be moved along with the entire entity tree contained in the entity being moved (*see chapter 3.4.3 Moving records in the classification scheme (re-classification)*).

*Warning: All these operations are only possible if the user has been given adequate access rights to perform them. Administrators can manage access rights at any time following the creation of the classification hierarchy.*

Users with adequate access rights are also allowed to do the following at any time following the creation of classes:

- Edit class metadata.
- Close classes, which will prevent:
  - The adding of new entities in the closed class or in the hierarchy below it
  - Changing archive content anywhere in the hierarchy below the closed class, including the metadata of the class itself.

### 3.4.3 Moving records in the classification scheme (re-classification)

No matter how carefully designed a classification scheme is, in the life cycle of an electronic archive the need inevitably arises to change the scheme.
Many times, it is necessary to change the classification code of a specific folder or document.
For this purpose, IMiS®/ARChive Server offers the possibility of moving an entity of any type to another location in the archive.
Moving an entity that contains a hierarchy with other entities causes the movement of the entire hierarchy to a different location.

 

The following conditions must be met to successfully move an entity in the archive:

- The user must have the right to read the entity being moved *(see chapter 3.3.5.2.1 Access rights for an entity)*.
- The user must have the right to move the entity being moved *(see chapter 3.3.5.2.1 Access rights for an entity)*.
- The user must have the right to create new entities in the location to the selected entity is being moved to *(see chapter 3.3.5.2.1 Access rights for an entity)*.
- The template of the entity being moved must suit the templates designated as allowed in the location the user is moving the entity to.
- The parent entity cannot have Closed status.
- The hierarchy the entity is being moved to must not be closed.
- Automatic classification codes must be enabled for all entities in the hierarchy under the entity being moved.
- The security class of the moving entity should be lower or equal to the security class of an entity, where user is about to move a moving entity.

Once an entity or a hierarchy of entities has been successfully moved, all moved entities are assigned new classification codes that conform to the coding rules of the new location.

Additional metadata is also added to the umbrella entity of the moved entities:

- sys:MoveReason: The reason for moving the entity; the user is required to provide this information when an entity is moved.
- sys:MoveAgent: The user who performed the move.
- sys:MoveDateTime: The date and time of the move (end).
- sys:MoveClassificationCode: The full classification code of the entity before the move.

For a detailed explanation of the metadata listed above *see chapter 3.2.5 System attributes*.
An event is recorded in the audit trail for every moved entity, for a description *see chapter 3.3.7.9 Move*.

 

When an entity movement event occurs (with an entity and its child entities), a report is created in the audit trail with the following information:

- Current status of the entity.
- Old and new value of the full classification code of the entity.
- Reason for the move; this information is entered by the user who moved the entity.
- Values of all metadata the entity contained when the move occurred.

## 3.5 Search

One of the most important functions of IMiS®/ARChive Server is the ability to search archived records and display folders and documents in line with a user's access rights.
A user can search for all types of entities using individual metadata values, key words and/or using the full text of documents stored in the archive. Users also have the option of recursive searching and searching by inherited values in the case of attributes whose values are inherited in line with their settings (security class, for example).

### 3.5.1 Data protection and security when searching

Search results display only classes, folders and documents for which a user has access rights either through the security or through the Access Control List (ACL). Entities for which the user does not have access rights will remain hidden to the user even if they meet the search criteria.
A user's other rights for operations on the entities in the search results are equal to their rights when normally viewing the archive.

*Example: If a user does not have the right to open an entity in Read mode, they will only see that it exists and will only be able to view its public metadata in the search results.*

### 3.5.2 Search syntax rules

A simple condition is the basic part of search syntax. IMiS®/ARChive Server use two types of simple conditions:

- Conditions for searching by metadata.
- Conditions for searching by full document text.

A search syntax consists of at least one simple condition.
Advanced search syntax consist of multiple simple conditions joined in the search syntax by logical operation (AND, OR, exclusive OR and negation).

 

Brackets can be used to further define the sequence of logical operations.

#### 3.5.2.1 Simple conditions for searching by metadata

IMiS®/ARChive Server enables users to search by all metadata for which searching has been enabled in the settings (searchable). Searching is available for all types of metadata.
A simple condition for searching by metadata consists of three components:

- The name of the metadata for which the condition is valid; in the condition, the name of the metadata is always listed in brackets.
  *Example: [Amount]*
- Comparison operation; The following operations can be used for comparison in simple conditions: less than (**<**), less than or equal to (**<=**), equal to (**=** or **==**), different (**<>** or **!=**), greater than or equal to (**>=**) and greater than (**>**).
- The value to which the comparison applies; All values can be listed in double quotes. If a value contains a space, the use of double quotes is required .

  *Example: "Lower Manhattan".*
  A period (.) is used to separate whole numbers from decimal values. Date and time values must be listed using XML notation. With time and date values, four special expressions that replace the whole value or one of its parts can be used.

  *@NOW@* the entire value will consist of current date and time
  *Example: [StartSession] < @NOW@*

  *@TODAY@* the date in the value will correspond to today's date
  *Example: [TimeReceived] < @TODAY@T08:00+02:00*

  *@YEAR@ the year in the value will correspond to the current year*
  *@MONTH@ the month in the value will correspond to the current month*
  *Example: [TimeImage] < @YEAR@@MONTH@-15T20:00+02:00*

  A special value for search criteria is the expression NULL. These values can be used to condition the existence of the values of specific metadata.
  When using the value NULL and with metadata of a logical kind, only the same as (== or =) and different (<> or !=) comparative operators can be used.

 

- Optionally, the arithmetic operations of addition and subtraction (+ or -) are stated after the name of metadata, where the second operand is a constant value.
  This option does not apply to sign and logic metadata.
  A special form of constant value is assigned for date metadata, consisting of at least one or all three components that represent years, months and days.
  Individual components consist of a whole number and a letter that represents a date unit as in the following examples:
  - use y or Y for years (e.g. 15y)
  - use m or M for months (e.g. 3m)
  - use d or D for days (e.g. 60d).

*Examples: Simple conditions for searching by metadata:*

*[Amount] - 40.5 >= 250*

*[Author]="John Smith"*

*[SessionStart] + 1y3m <= 2014-05-07T11:05+02:00*

*[EndStart] + 7d <= @NOW@*

*[Author]!=NULL*

#### 3.5.2.2 Simple conditions for searching by full text of content

IMiS®/ARChive Server enables users to search using the full text of content for which searching has been enabled in the settings. This search option is available for content of all types.

Simple conditions for searching by full text of content are listed in curly in search syntax: {"John Smith"}. Within the curly brackets, a simple full text search term can be expanded to a more complex full text search term using logical operations.
The following logical operations are supported for terms for searching by full text of content; they are listed in the order in which they are calculated:

- Negation: NOT .
- And: AND .
- Or: OR .

In the full text search terms, parentheses can be used to change the order in which operations are performed.

 

*Example: {"Smith" AND "John" OR "Frank"} will find all texts where both words, "Smith" and "John", appear and texts where the word "Frank" appears.*

*Example: {"Smith" AND ("John" OR "Frank")} will find all texts where the word "Smith" appears in combination with either the word "John" or the word "Frank".*

#### 3.5.2.3 Logical operations in search syntax

Simple terms link to form complex terms using logical operations.
IMiS®/ARChive Server supports the following logical operations; they are listed in the order in which they are performed:

- Negation: NOT or ! .
- And: AND or & .
- Excluding or: XOR or ^ .
- Or: OR or | .

*Example: The term:[A]>3 OR [A]=3 AND [B]!=NULL is equal to the term: NOT [B]== NULL & [A]==3 | [A]>3*
*In both cases, the "and" operation is performed first, and then the "or" operation.*
*In the second example, the negation operation "NOT" is performed before "AND".*
Parentheses can be used to change the default order in which logical operations are performed. Using parentheses, the search term shown above becomes something different, and the search results will also be completely different.

*Example: The use of parentheses causes the "OR" operation to be performed before the "AND" operation: ([A]>3 OR [A]=3) AND [B]!=NULL*

Both types of simple terms can also be used in the same search syntax.
*Example: The term: [Surface] >= 12 AND {"John AND Smith"} is correctly formed.*

We recommend joining full text search terms in a single term in curly brackets if possible.
*Example: The term:[Surface] >= 12 AND {"John"} AND {"Smith"} will return the same results as the term [Surface] >= 12 and {"John" AND "Smith"}*
*The search will be slightly faster with the second term.*

 

## 3.6 Authenticity

### 3.6.1 Conditions

The metadata and content of entities may change throughout their life cycle.

A precondition for ensuring the authenticity of entities is their inalterability.

IMiS®/ARChive Server uses the recommendations of the LTANS to mathematically check integrity. LTANS (Long-Term Archive and Notary Services) is a body that prescribes de facto standards for long-term content archiving procedures. Entities are included in the procedure according to the rules described below (*see chapter 3.6.7 Rules*). It is recommended that rules are written in such a way that only unchangeable entities (i.e. closed entities) are included in the authentication process. Since the content of the entities can later be changed, proofs for such entity are generated repeatedly.

Additionally, an entity must have at least one metadata that denotes it is an object of an Archival Information Package – AIP.

AIP is an XML representation of the content of the entity and its metadata, and is described in more detail below.

The authenticity of an AIP and consequently of the entity to which it pertains is proven by the ERS created in the process of determining the authenticity of data subject to long-term archiving.

The illustration below shows the process of including entities that meet the conditions for the process for determining the authenticity of data subject to long-term archiving.

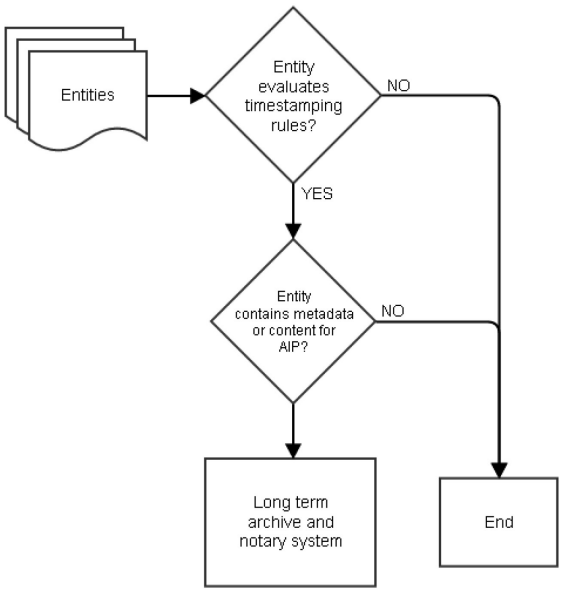


Image 7: Entity role in the process of keeping the authenticity of data for long-term archiving

 

### 3.6.2 Concept

One of the key concepts of a secure digital archive is maintaining authenticity of the archived records for the entire duration of their storage.
This means that the original properties of the document must be preserved with regard to its context, structure and content. The key is that from the moment a document is archived all parts (structure, content, metadata, etc.) necessary for ensuring authenticity are no longer changed. The authenticity of electronically archived records can be determined using the following functionalities:

- Hashing - #.
- Digital signatures with digital certificates.
- Timestamps.

#### 3.6.2.1 Hashing - #

A hash is the result of a one-way hashing functions that takes a block of data as input (for example, the string of bits of the binary content of a document, the original text of a document) and calculates a so-called hash with a fixed length. Even the slightest change in the input data will result in an enormous change in the value of the hash.

The properties of an ideal hashing function are as follows:

- A hash can easily be calculated for every input block of data.
- The input block of data cannot be ascertained from the hash (the hashing function only works one way).
- It is impossible to alter the input data in such a way that the hash would remain unchanged.
- It is theoretically impossible for the hashes of different blocks of data to be identical or collide.

Technological development and gains in processing power have led to an increase in the likelihood of a collision of hash values, which weakens the security of the hashing function. Today, for example, the MD5 hash function can now be cracked and as such is no longer safe for use.

 

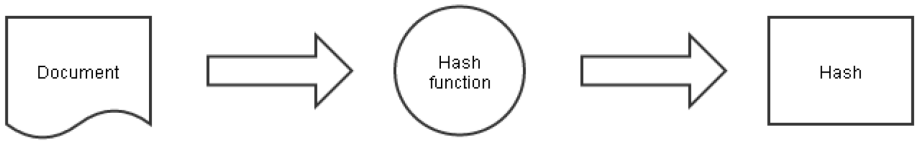


Image 8: How the hashing function works

### 3.6.2.2 Digital signature and digital certificate

A digital signature is based on a PKI or public key infrastructure and mathematically ensures the authenticity of a digital document.

The digital signature scheme is based on three algorithms:

- Key generation: A private and public key are generated; these keys are crucial for generating and verifying a signature.
- Signature algorithm: A digital signature is generated on the basis of the private key and the document.
- The signature verifying algorithm: The authenticity of the document is checked on the basis of the digital signature and the public key.

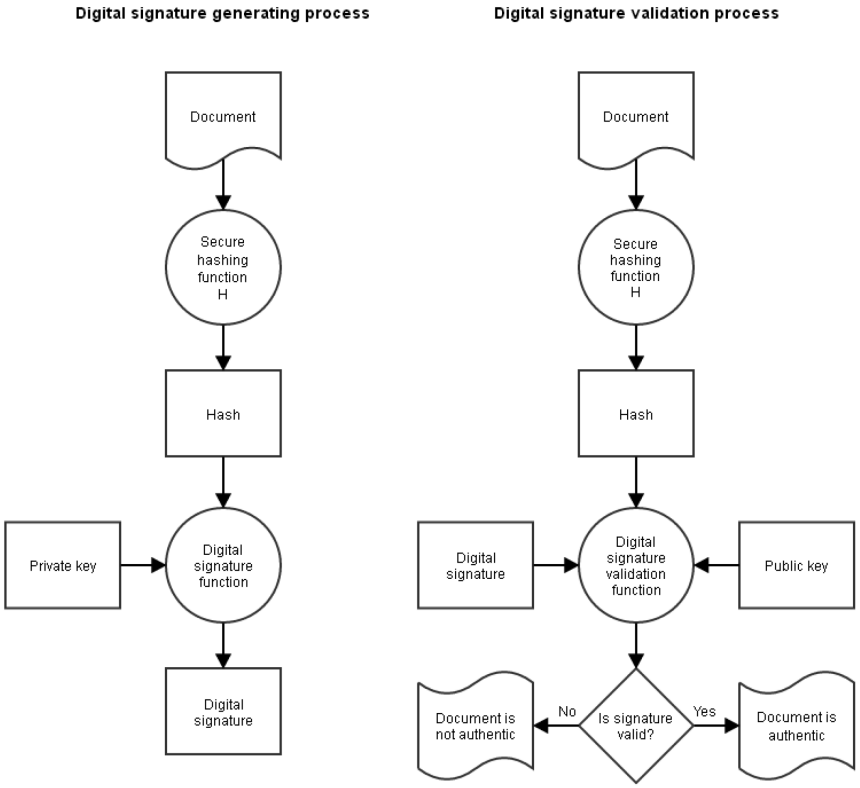


Image 9: Generating and verifying a digital signature

The validity of a digital signature depends on the validity of the digital certificate.
To preserve the authenticity of a document, it must be signed with a new digital certificate before the old one expires.

#### 3.6.2.3 Timestamps

The timestamping process is equivalent to digital signing, except that it also involves a TSA or Time Stamping Authority.
The latter adds a time component to the digital signature. This time component or timestamp uniquely specifies the time of the digital signature of the document. As in the case of digital signatures, the validity of timestamps is limited by the validity of the digital certificate.
As noted above, a digital certificate needs to be renewed to ensure document authenticity.
A timestamp is used to renew digital signatures.

It is a practical solution, especially if content has been signed by multiple persons whose digital certificates will eventually expire. Without a timestamp, all the signatories whose digital certificates had expired would need to sign the document again in order to ensure the authenticity of the content.
With a timestamp, the authenticity of the content can be ensured even if the digital certificates of the signatories have expired.

All the methods listed above (hashing, digital signatures and certificates, timestamps) have their shortcomings, both in terms of technology (hashes can potentially be cracked) and duration (digital certificates expire). That is why the use of just one of these methods is not suitable to ensure the long-term authenticity of a document.
There exist a number of standards that use a combination of hashing, digital signatures and timestamps to ensure the long-term authenticity of documents:

- ETSI standards.
- Evidence Record Syntax – ERS.
- Auditing Control Environment – ACE.
- Content Integrity Service – CIS.

IMiS®/ARChive Server use the ERS standard in XML form (XMLERS, RFC 6283) to provide for the long-term authenticity of documents. This standard is described below.

 

### 3.6.3 Storing digital certificates

IMiS®/ARChive use a certificate store for work with digital certificates. It is based on the open source OpenSSL library.

Storing digital certificates enables the following:

- Adding digital certificates.
- Removing digital certificates.
- Searching digital certificates.
- Searching a list of revoked digital certificates.

To speed up the process of obtaining digital certificates and current information on revoked certificates, the store caches the currently used certificate chain and information about revoked certificates.

#### 3.6.3.1 The digital certificate chain

The digital certificate chain is a list of digital certificates that proves the authenticity of an individual digital certificate. Every digital certificate in the list is signed with the private key of the next digital certificate in the list.

At the end of the list is the certificate authority root certificate, which is a self-signed certificate. This kind of chain is known as a chain of trust and serves to ensure the authenticity of the certificates it contains.

The illustration below shows an example of a certificate chain with three digital certificates and the process of verifying individual certificates:

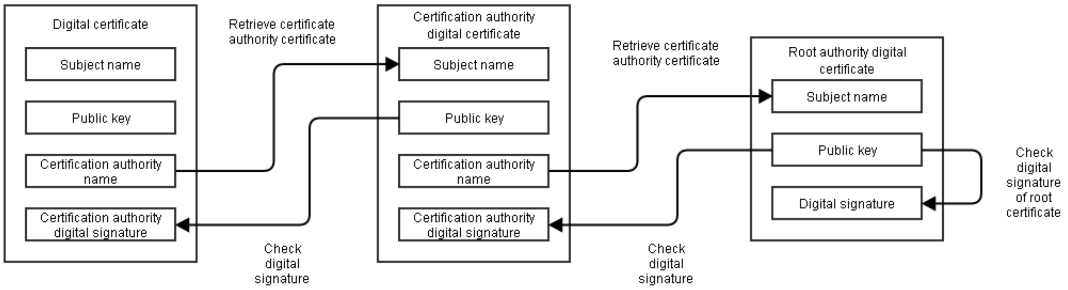


Image 10: The digital certificate chain

### 3.6.3.2 Adding digital certificates

Digital certificates will be successfully added to the certificate store if the following conditions are met:

- The digital certificate must be valid at the time it is added to the store (i.e. cannot be expired).
- The store must contain all previous digital certificates that make up the chain of trust.
- If you are concerned about whether a digital certificate has been revoked when adding the certificate or about the potential revocation of the other digital certificates, note that adding will succeed only if a check reveals that none of the certificates have been revoked.

There are two mandatory conditions for adding certificates to a store:

- The certificate is valid at the time of adding.
- There are already certificates in the store that make up the chain of trust.

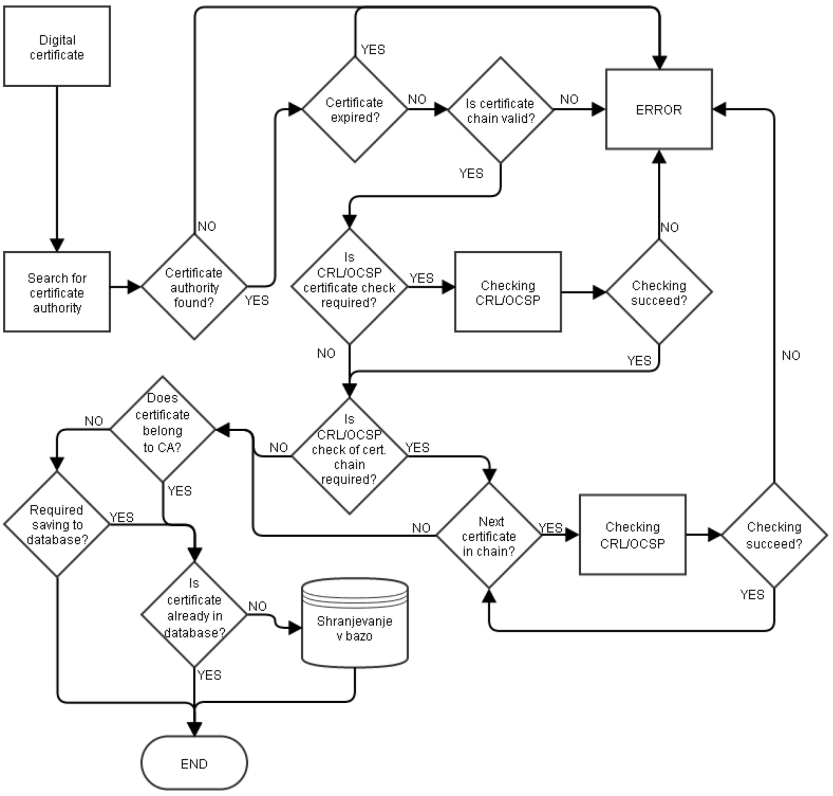


Image 11: Adding a digital certificate to a store

#### 3.6.3.3 Removing digital certificates

The removal of a digital certificate from the store is only possible if the digital certificate has not signed any other digital certificates. Also, the certificate you would like to remove has to be located at the end of the chain. Once successfully removed, the digital certificate is removed from the database and the cache.

#### 3.6.3.4 Searching digital certificates

Digital certificates can be searched for in the store using their hash (the SHA-512 hashing algorithm is used) or using the unique 32-bit identifier they are assigned when saved in the database. Digital certificates not stored in the database do not have an identifier.

#### 3.6.3.5 Searching information about revoked certificates

Information about revoked certificates can be searched for using the 64-bit unique identifier assigned by the server when a certificate is saved in the database. The saving of information on revoked certificates is performed when verifying timestamps in the process for determining the authenticity of data subject to long-term archiving *(see chapter 3.6.4 ERS and chapter 3.6.6 Time stamping)*.

Every digital certificate has so-called certificate extensions. These extensions are described in the RFC 5280 specification (*http://tools.ietf.org/html/rfc5280*).

Extensions contain information about distribution points where information about the following can be obtained:

- Revoked digital certificates.
- Information about what type a digital certificate is.
- Information about the intended uses of a digital certificate.
- Other information.

Besides information about distribution points, when processing digital certificates, the server also uses extensions called basic constraints and extended key usage.

#### 3.6.3.6 Basic constraints

The extension contains information about whether a digital certificate belongs to a certificate authority or not. The information is recorded in the “mCACertificate” parameter of the digital certificate and can have a value of “1” or “0”.

A value of “1” means that the digital certificate belongs to a certificate authority, and a “0” means that it does not.

 

#### 3.6.3.7 Extended key usage

An extension contains parameters with information about the purpose of the usage of a digital certificate.

Parameter: mServerAuthentication
Valid values: 1 or 0
Description: If the value is 1, the digital certificate can be used to authenticate the server when setting up an encrypted connection between server and client.

Parameter: mClientAuthentication
Valid values: 1 or 0
Description: If the value is 1, the digital certificate can be used to authenticate the client when setting up an encrypted connection between server and client.

Parameter: mEmailProtection
Valid values: 1 or 0
Description: If the value is 1, the digital certificate can be used to sign email messages.

Parameter: mCodeSigning
Valid values: 1 or 0
Description: If the value is 1, the digital certificate can be used to sign executable code.

Parameter: mMicrosoftServerGatedCrypto
Valid values: 1 or 0
Description: This parameter is not used.

Parameter: mNetscapeServerGatedCrypto
Valid values: 1 or 0
Description: This parameter is not used.

Parameter: mOcspSign
Valid values: 1 or 0
Description: If the value is 1, the digital certificate can be used to sign OCSP server replies that transmit information about revoked digital certificates.

 

Parameter: mTimestamp

Valid values: 1 or 0

Description: If the value is 1, the digital certificate can be used for timestamping.

Parameter: mDvcs

Valid values: 1 or 0

Description: If the value is 1, the digital certificate can be used to sign components in the DVCS protocol (the protocol is described in the RFC 3029 specification- *http://tools.ietf.org/html/rfc3029*).

### 3.6.4 ERS

Evidence Record Syntax – ERS is the standard that describes the system for ensuring the authenticity of data subject to long-term archiving.
ERS prescribes how authenticity proofs are to be created and renewed. At the same time, it describes how they are to be structured in order to unambiguously prove the authenticity of the archived records from the moment they enter the process for ensuring long-term authenticity.
IMiS®/ARChive Server performs ERS in XML format using the RFC 6283 standard (*https://tools.ietf.org/html/rfc6283*).

The key processes for ensuring the long-term authenticity of data are:

- The process for generating proofs.
- The process for renewing proofs.

Before beginning the process for generating authenticity proofs, the content and metadata the evidence syntax will be used to secure or prove must be selected. This is done by creating an Archival Information Package (AIP) for every entity that meets the conditions for ensuring the long-term authenticity of stored data *(see chapter 3.6.1 Conditions)*.

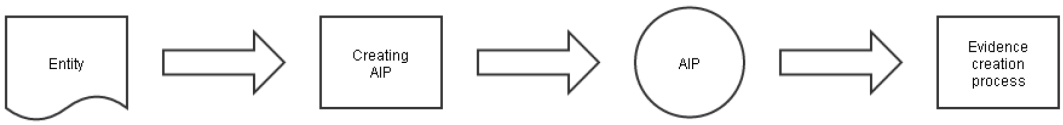


Image 12: Creating an AIP for generating authenticity proofs

 

The authenticity proofs (hashes, digital signatures, timestamps) have a limited life span.
The validity of a digital signature and timestamp are limited by the validity period of the digital certificate used to create them.
That is why they must be renewed (a new digital signature or timestamp must be generated).
This is what is known as the simple authenticity proof renewal process.

Over time, the degree of security provided by the algorithm for creating hashes diminishes, as the likelihood of collisions between hashed values grows. That is why the algorithm must be replaced with a new one. This is what is known as the complex authenticity proof renewal process. It is used to ensure that the proofs do not become unreliable even though the underlying mathematical algorithms deteriorate, so to speak, over time.

Generated authenticity proofs must be periodically checked and renewed before they become invalid, as shown in the illustration below.

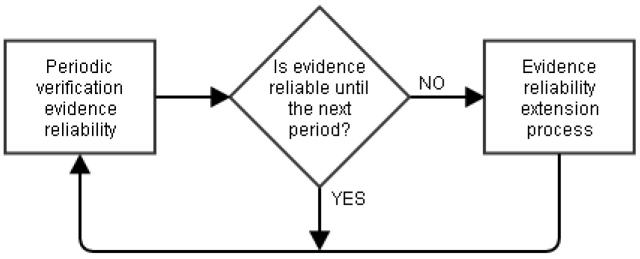


Image 13: The authenticity proof renewal process

Verification of the authenticity of archived records takes place in two steps:

- Checking the AIP against data in the ERS.
- Checking the content and metadata of the entity against the values in the AIP.

If both checks are successful, that means that the content and metadata of the entity have remained unchanged for the entire duration of their storage. If any of the checks are unsuccessful, that means that the data cannot be guaranteed to be authentic.

 

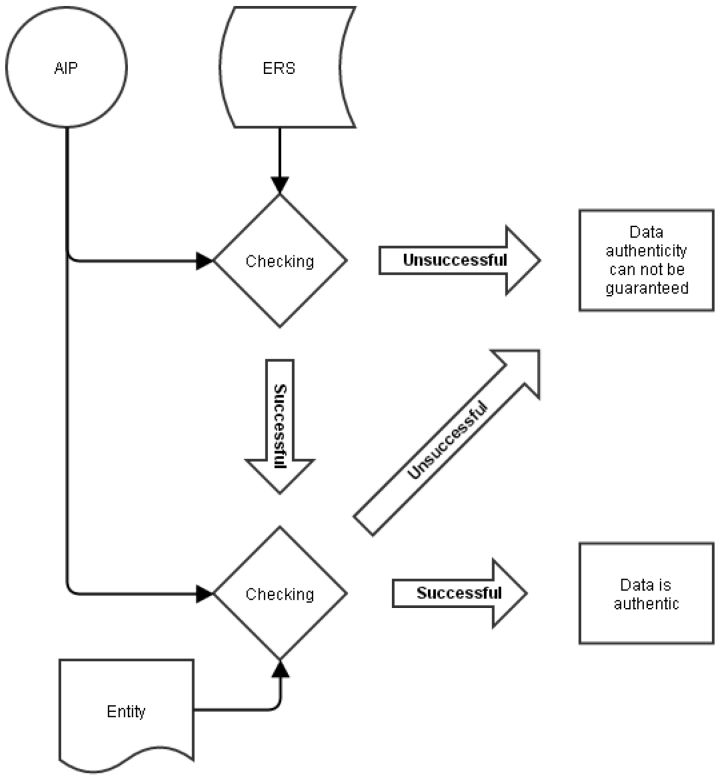


Image 14: Example of verifying data authenticity

#### 3.6.4.1 The proof generating process

For every entity that enters the authenticity proof process for the long-term archiving of data, an Archival Information Package (AIP) is created. This package is processed by IMiS®/ARChive Server on the basis of the information found in the package header (*see chapter 3.6.5 AIP*).

A hash is then calculated for the processed package; it serves as the basis for timestamping. The timestamp unambiguously proves that the hash (and the AIP to which it belongs) existed before the time listed in the timestamp.

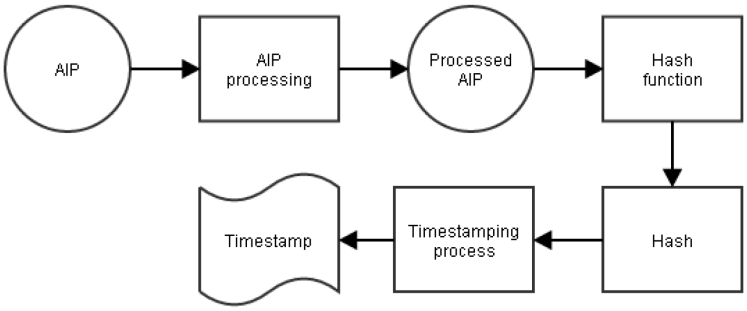


Image 15: The time stamping process for an individual AIP

 

Because the timestamping process is rather demanding, and most timestamp providers charge for their services, it would not be rational to timestamp individual hashes.
That is why timestamping is usually done with packages. This involves the following steps:

- AIPs are created for all entities for which proof of authenticity needs to be created.
- All created AIPs are processed by IMiS®/ARChive Server, and hashes are generated from them.
- These hashes are used to construct a hash tree - the server then uses so-called Merkle trees for timestamping *(see chapter 3.6.4.3 Merkle tree)*.
- The Merkle tree is then used to calculate a root hash that secures the entire tree of hashes.
- The root hash is timestamped, creating proof of the existence of all hashes (and their AIPs) in the Merkle tree before the time listed in the timestamp.

A Merkle tree can be used to timestamp a package with a large number of entities and therefore considerably rationalizes the authenticity proof generation process.
Once proofs have been generated, their validity must be maintained through renewals.

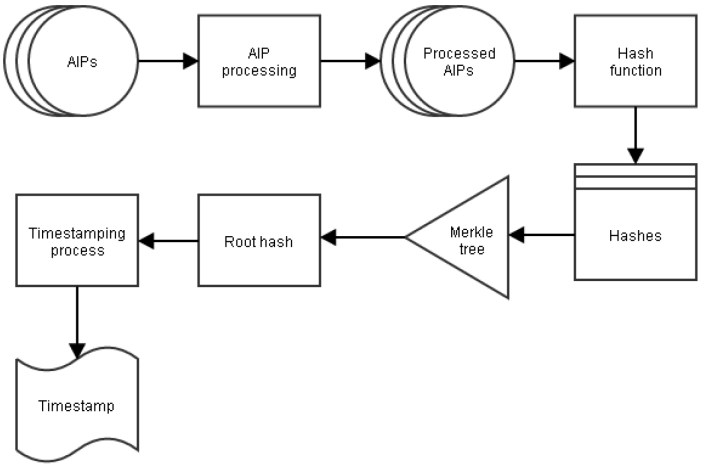


Image 16: The time stamping process using a Merkle tree

### 3.6.4.2 The authenticity proof renewal process

All generated authenticity proofs must be renewed before their validity expires, as otherwise they cannot be used to ensure the authenticity of long-term data archiving.
The renewal of authenticity proofs can be broken down into two methods:

- Simple renewal.
- Complex renewal.

 

#### 3.6.4.2.1 Simple authenticity proof renewal process

Simple authenticity proof renewal process is performed when the digital certificate that created the timestamp is about to expire. The timestamp is renewed simply by calculating and then timestamping its hash. The new timestamp extends the reliability of the old one even if the digital certificate associated with the old stamp has expired. This is called a timestamp chain.

Because it would be imprudent to renew individual timestamps, Merkle trees are used in line with the same process used for AIPs. The simple authenticity proof renewal process is combined with the proof generation process.
A Merkle tree is used; through the hashing process, AIP hashes and timestamps that are about to expire are added.
By timestamping the root hash of the Merkle tree, authenticity proof is created of the existence of the AIP and the validity of the timestamps that are about to expire is renewed.

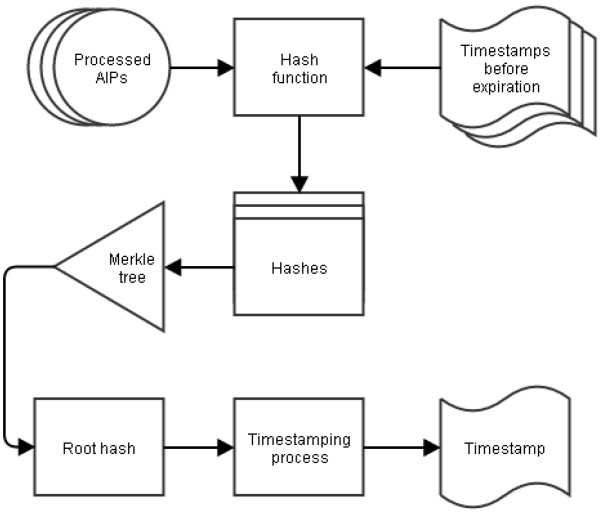


Image 17: The simple authenticity proof renewal process in combination with proof generation

#### 3.6.4.2.2 Complex authenticity proof renewal process

Complex authenticity proof renewal process is performed when the deterioration of the degree of security of a hashing algorithm used to create AIP hashes is foreseen. In this case, the following steps must be used to renew hashes before the hashing algorithm becomes weak.

1. A new, secure hashing function is selected.
2. A new hash is calculated for the AIP.
3. A new hash is calculated from all proofs that belong to the AIP in step 2.
4. The hashes from steps 2 and 3 are concatenated, and their new joint hash is added to the Merkle tree *(see chapter 3.6.4.3 Merkle Tree)*.

 

5. The steps described in 2, 3 and 4 must be performed for every AIP that was hashed using the potentially weak algorithm.
6. A root hash is then calculated for the Merkle tree.
7. The root hash is time stamped.

Complex authenticity proof renewal is used to ensure the reliability of an AIP and all the proofs associated with it. The new timestamp is the start of a new timestamp chain which can be renewed using simple authenticity proof renewal as long as the hashing function used in the last complex authenticity proof renewal is considered secure.

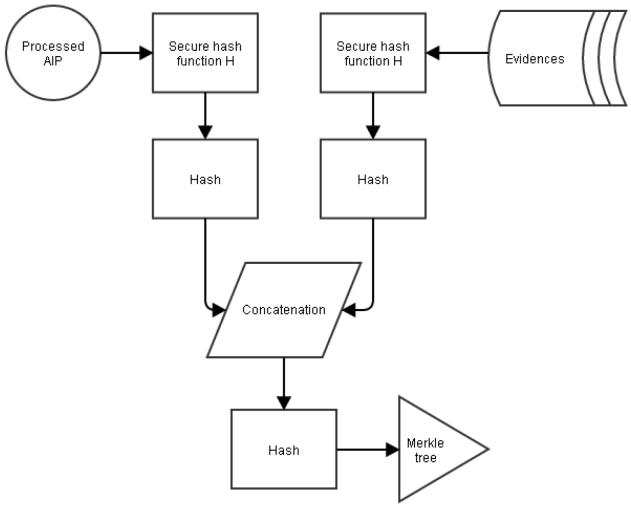


Image 18: The part of the complex authenticity proof renewal process (steps 2, 3, 4)

#### 3.6.4.3 Merkle tree

A Merkle tree is a tree-shaped data structure in which the nodes of the tree are hashes created using the same type of hashing function. The tree has the following properties:

- It contains only hashes created with the same type of hashing function
  (a Merkle tree cannot contain hashes created with the MD5 and hashes created with the SHA1 algorithm, for example).
- Hashes are added to the tree as leaf nodes.
  These are nodes without child nodes.
- Adding and removing hashes is only possible before the tree is constructed.
- The root hash is the hash calculated when the tree is constructed; it secures the entire tree.
- The constructed tree can be reduced only to those nodes that are required to calculate the root hash from an individual leaf node.

 

IMiS®/ARChive Server constructs Merkle trees using a binary tree. A binary tree is a data structure with a node that can have a maximum of two child nodes.

#### 3.6.4.3.1 Constructing the Merkle tree

The construction of a Merkle tree is the process of constructing nodes from the leaf node to the root node.

Node construction follows these steps:

- Two hash values are taken (from leaf nodes).
- The binary values of the hashes are sorted by size from smallest to largest.
- The sorted values are joined or concatenated.
- The hashes of the value sorted in this way are used to calculate a new hash of the same type that represents a new node in the tree.

This process can be repeated as long as more than one node is available. When only one node remains the tree is constructed. The node that remains represents the root hash.

*Example: A Merkle tree with four leaf nodes with hashes H1, H2, H3 and H4. Let's assume the following relations exist between the values of the hashes:*

- *The binary value of H1 is less than that of H2 (H1 < H2).*
- *The binary value of H3 is greater than that of H4 (H3 > H4).*

On the basis of these relations the following formulas can be used to calculate two new nodes, H12 and H34:

- H12 = HASH (H1 || H2) – because H1 < H2, the sorting maintains the order, which is why the two hashes are only joined (the || operation)
- H43 = HASH (H4 || H3) – because H3 is greater than H4, the sorting reverses the order of the hashes.

Let's assume the following relations exist between nodes H12 and H43: H12 < H43.
The following node is calculated from these two hashes: H1243 = HASH (H12 || H43).
Because H1243 is the last node in the tree, the tree has been constructed; the value of this node is therefore the root hash.

 

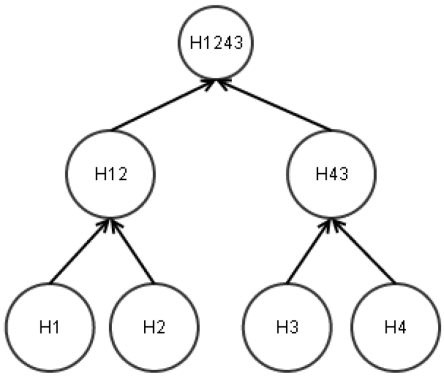


Image 19: Example of a Merkle tree

#### 3.6.4.3.2 A reduced Merkle tree

The constructed tree can be reduced only to the number of nodes that are required to calculate the root hash. Reduction follows these steps:

- A leaf node is selected; this node will be used to create a reduced Merkle tree.
- Let's take a look at the parent node of the selected node; we will take all nodes directly under this node (one of which is the node we selected in the previous step).
  The values of the hashes in the nodes are sorted in ascending order (similarly to how nodes are constructed).
- We continue by searching for all nodes directly above it, until there are no more (meaning we have come to the root node). All nodes found in this way are sorted in ascending order.
- The nodes directly above it are not included in the reduced tree, as they can be calculated in accordance with the rules for constructing a Merkle tree.

*Example: The reduced Merkle tree we used to construct a tree.*
*Let's select leaf node H1. The node directly above it is H12, and its child nodes are H1 and H2.*
*We take both nodes and sort them by the size of their hashes (let's assume H1 < H2). Node H1243 is directly above node H12, and its child nodes are H12 and H43.*
*Because node H12 can be calculated from nodes H1 and H2, it is left out and only node H43 is included in the reduced tree. Node H1243 does not have a node directly above it, which is why the reduction of the tree is finished for node H1.*

The reduced tree therefore contains nodes in the following order: H1, H2, H43.
As the illustration below shows, the reduced tree for node H2 is the same, as H1 and H2 have the same parent node, H12

 

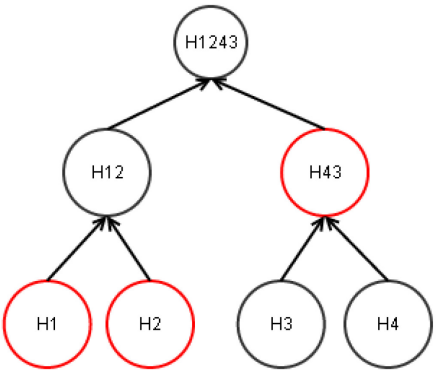


Image 20: An example of a reduced tree for nodes H1 and H2

*An example of a root hash:*

*The root hash of the reduced tree is calculated the following way:*

*H1243 = HASH (HASH (H1 || H2) || H43).*

*The calculation process is as follows:*

- *H12 = HASH (H1 || H2) – the intermediate node H12 is calculated.*
- *SORT (H12, H43) = [H12, H43] – the sorting results in H12 and H43 (H12 < H43).*
- *H1243 = HASH (H12 || H43) – the root hash is calculated.*

*Let's look at an example of the reduced tree for nodes H3 and H4, which contains the following order of nodes: H4, H3, H12.*

*The root hash is calculated the following way:*

*H1243 = HASH (H12 || HASH (H4 || H3) ).*

*The process is as follows:*

- *H43 = HASH (H4 || H3) – the intermediate node H43 is calculated.*
- *SORT (H43, H12) = [H12, H43] – the sorting results in H43 and H12 (H12 < H43).*
- *H1243 = HASH (H12 || H43) – the root hash.*

#### 3.6.4.4 Syntax

As noted elsewhere, IMiS®/ARChive Server performs ERS in XML format.

##### 3.6.4.4.1 "EvidenceRecord" tag

The root XML element is the EvidenceRecord tag, which contains the elements listed in the table below.

 

| Name of the XML element | Type of the XML element | Required |
|---|---|---|
| Version | Attribute | YES |
| ArchiveTimeStampSequence | Tag | YES |

Table 7: XML elements of the EvidenceRecord tag

"Version" attribute

This attribute has the fixed value "1.0" and represents the ERS version.

"ArchiveTimeStampSequence" tag

The content of this tag is the sequence of all archived timestamp chains, which shows the generation and renewal of authenticity proofs of entities stored for extended periods of time (*see chapter 3.6.4.1 The proof generating process* and *chapter 3.6.4.2 The authenticity proof renewal process*).

Every timestamp sequence is marked with the tag ArchiveTimeStampSequence.

*Example: An XML record representing an ERS with two chains of timestamps.*

```
<EvidenceRecord xmlns="http://www.setcce.org/schemas/ers" version="1.0">
    <ArchiveTimeStampSequence>
    <ArchiveTimeStampChain order="1">...</ArchiveTimeStampChain>
    <ArchiveTimeStampChain order="2">...<ArchiveTimeStampChain>
    </ArchiveTimeStampSequence>
</EvidenceRecord>
```

#### 3.6.4.4.2 "ArchiveTimeStampChain" tag

This label contains a chain of archived timestamps created in processes for creating and renewing using the same hashing function.

The elements of the tag are described in the table below.

| Name of the XML element | Type of the XML element | Required |
|---|---|---|
| Order | Attribute | YES |
| DigestMethod | Tag | NO |
| CanonicalizationMethod | Tag | YES |
| ArchiveTimeStamp | Tag | YES |

Table 8: XML elements of the ArchiveTimeStampChain tag

 

"Order" attribute

The value of this attribute represents the sequence of timestamp chains.

When the first archive timestamp is created, the construction of a timestamp chain with the attribute value 1 begins. All archive timestamps created with simple renewal are then added to this chain.

For every complex authenticity proof renewal process, a new archive timestamp chain is created with the attribute value 2, 3 and so on *(see chapter 3.6.4.2 The authenticity proof renewal process)*.

"DigestMethod" tag

This tag contains the required Algorithm attribute. The value of the attribute is a URI that prescribes the hashing algorithm used in the chain. This label is not required, as information about the hashing algorithm is also found in the individual timestamps.

"CanonicalizationMethod" tag

This tag contains the Algorithm attribute. Its value is the URI or Uniform Resource Identifier that prescribes the algorithm for converting a chain to a canonicalized chain.

#### 3.6.4.4.2.1 "ArchiveTimeStamp" tag

This archive timestamp contains the timestamp (with all data required for its verification) and a reduced Merkle tree if multiple AIPs are time stamped. The elements of the tag are described in the table below.

*Example: The XML report shows a chain containing three timestamps generated using the SHA1 hashing algorithm.*

```
<ArchiveTimeStampChain order="1">
    <DigestMethod algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <CanonicalizationMethod algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <ArchiveTimeStamp order="1">...</ArchiveTimeStamp>
    <ArchiveTimeStamp order="2">...</ArchiveTimeStamp>
    <ArchiveTimeStamp order="3">...</ArchiveTimeStamp>
</ArchiveTimeStampChain>
```

 

| Name of the XML element | Type of the XML element | Required |
|---|---|---|
| Order | Attribute | YES |
| HashTree | Tag | NO |
| TimeStamp | Tag | YES |

Table 9: XML elements of the ArchiveTimeStamp tag

"Order" attribute

The value of this attribute represents the order of an archive timestamp within the chain.
An archive timestamp with the value 1 was created using the proof generation process, and all other archive timestamps were created in the simple authenticity proof renewal process.

"HashTree" tag

If this tag is present, it represents a reduced Merkle tree above which an archive timestamp was created. If the tag is not present, the archive timestamp belongs to an object that has a hash in the timestamp in the TimeStamp tag.

This tag represents a reduced Merkle tree. It contains an order of Order elements that represent the levels of the Merkle tree. The first level (level 0) is not included because the value of the root node can be calculated. The elements of the Sequence tag are described in the table below.

| Name of the XML element | Type of the XML element | Required |
|---|---|---|
| Order | Attribute | YES |
| DigestValue | Tag | YES |

Table 10: XML elements of the Sequence tag

"Order" attribute

The value of this attribute represents the level of the reduced Merkle tree; levels are in reverse order compared the depth of the tree *(see chapter 3.6.4.3 Merkle tree)*.
The first valid value of this attribute is 1, which represents the hashes of the leaf nodes.
The value 2 represents the next level, and so forth.

"DigestValue" tag

The value of this tag contains a Base64 encoded hash.

 

*Example: An XML record shows a reduced Merkle tree with two levels.*
*The first order (order 1) contains Base64 coded hashes of the leaf nodes, and the second order contains the hash of the node that makes it possible to calculate the hash of the root node.*

```
<HashTree>
    <Sequence order="1">
        <DigestValue>D+/oVEs6CjRHi3UNL1vk4WcsEkA=</DigestValue>
        <DigestValue>EBEhfDzZh9+rfb1Kaqe65o7TTok=</DigestValue>
    </Sequence>
    <Sequence order="2">
        <DigestValue>fBuhe8txb0OmNt27uvYupJTrgBQ=</DigestValue>
    </Sequence>
</HashTree>
```

"TimeStamp" tag

This tag contains the timestamp and a list of cryptographic elements (the entire branch of digital certificates and information about revoked digital certificates) needed to verify the timestamp. This data can be used to unambiguously place the validity of the timestamp at the time it was made and consequently to prove the authenticity of the content it covers.

| XML element name | XML element type | Required |
|---|---|---|
| TimeStampToken | Tag | YES |
| CryptographicInformationList | Tag | NO |

Table 11: XML elements of the TimeStamp tag

"TimeStampToken" tag

This tag contains a timestamp obtained from an issuer of secure timestamps.
It is required to have the Type attribute, which specifies the type of the timestamp.

| Type | Description |
|---|---|
| XMLENTRUST | The timestamp is in the XML format prescribed by the W3C consortium (*www.w3.org/TR/xmldsig-core/*) and is the TimeStampToken content. |
| RFC3161 | The timestamp was made using the RFC3161 standard. The content of TimeStampToken is encoded in Base64 due to XML compatibility. |

Table 12: Supported timestamps

 

"CryptographicInformationList" tag

If this tag is present, it contains cryptographic elements that enable verification of the integrity of the timestamp. If this tag it not present, the timestamp itself contains these elements.
The Order and Type attributes are required for this tag; the latter specifies the type of cryptographic element.

| Type | Description |
|---|---|
| CERT | This cryptographic element is the Base64 encoded X509 digital certificate in binary form. |
| CRL | This cryptographic element is the Base64 encoded list of revoked digital certificates in binary form. |
| OCSP | This cryptographic element is the Base64 encoded server reply (in accordance with the protocol for continuous checking of the status of a digital certificate) in binary form. |

Table 13: Supported cryptographic element types

The example below shows the first archive timestamp that contains a reduced Merkle tree, a timestamp in XML format and its accompanying list of cryptographic elements.
The first in the list is the root digital certificate, which, together with the digital certificate from the timestamp, forms a chain of digital certificates.

The other element is the list of expired digital certificates, which can be used to check that the digital certificate that generated the timestamp is not expired.

```
<ArchiveTimeStamp order="1">
<HashTree>
    <Sequence order="1">
        <DigestValue>D+/oVEs6CjRHi3UNL1vk4WcsEkA=</DigestValue>
        <DigestValue>EBEhfDzZh9+rfb1Kaqe65o7TTok=</DigestValue>
    </Sequence>
    <Sequence order="2">
        <DigestValue>fBuhe8txb00mNt27uvYupJTrgBQ=</DigestValue>
    </Sequence>
</HashTree>
<TimeStamp>
    <TimeStampToken type="XMLENTRUST">
        <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">...</dsig:Signature>
```

 

```
    </TimeStampToken>
    <CryptographicInformationList>
        <CryptographicInformation order="1" type="CERT">MIIEHDCCAwS...
        </CryptographicInformation>
        <CryptographicInformation order="2" type="CRL">MIISKTCCEREC...
        </CryptographicInformation>
    </CryptographicInformationList>
</TimeStamp>
</ArchiveTimeStamp>
```

#### 3.6.4.5 Simple renewal

With simple renewal, only the last archive timestamp in the last archive timestamp chain is renewed.

The process is as follows:

1. The system checks if the last archive timestamp contains all the required cryptographic elements which prove its integrity (certificates, information about revoked digital certificates). If it does not contain these elements, the server automatically obtains them and adds them to the list of cryptographic elements.
   If proofs cannot be obtained, the process cannot be continued.
2. On the basis of the CanonicalizationMethod tag from the timestamp chain, an algorhythm is selected for conversion to a canonical form.
3. The TimeStamp tag is taken from the last archive timestamp and is canonicalized with the algorithm selected above.
4. The selected hashing function (specified in the DigestMethod tag in the current chain or contained in the timestamp being renewed) is used to calculate the hash of the canonicalized data. This is then timestamped (or added to the Merkle tree if multiple timestamps are being renewed or authenticity proof elements are being generated for a combination of timestamps). *(see chapter 3.6.4.2.1 Simple authenticity proof renewal process)*.
5. The timestamp is also checked *(see chapter 3.6.6.2 Verifying a timestamp)*.
6. If the check of the timestamp fails, the entire auhenticity proof renewal process must be repeated; otherwise it would not be possible to ensure the integrity of the new timestamp.

 

7. If the check is successful, a new ArchiveTimeStamp tag is created with an Order attribute with a value one value greater than the archive timestamp being renewed.
   If the archive timestamp was renewed using a Merkle tree, the reduced Merkle tree to which it belongs is included in the timestamp. In the last step, a TimeStamp tag is created. The verified timestamp is added to the tag, along with a list of cryptographic elements used in the verification process. The new archive timestamp is added to the last timestamp chain.

The illustration below shows how the simple archive timestamp renewal algorithm functions.

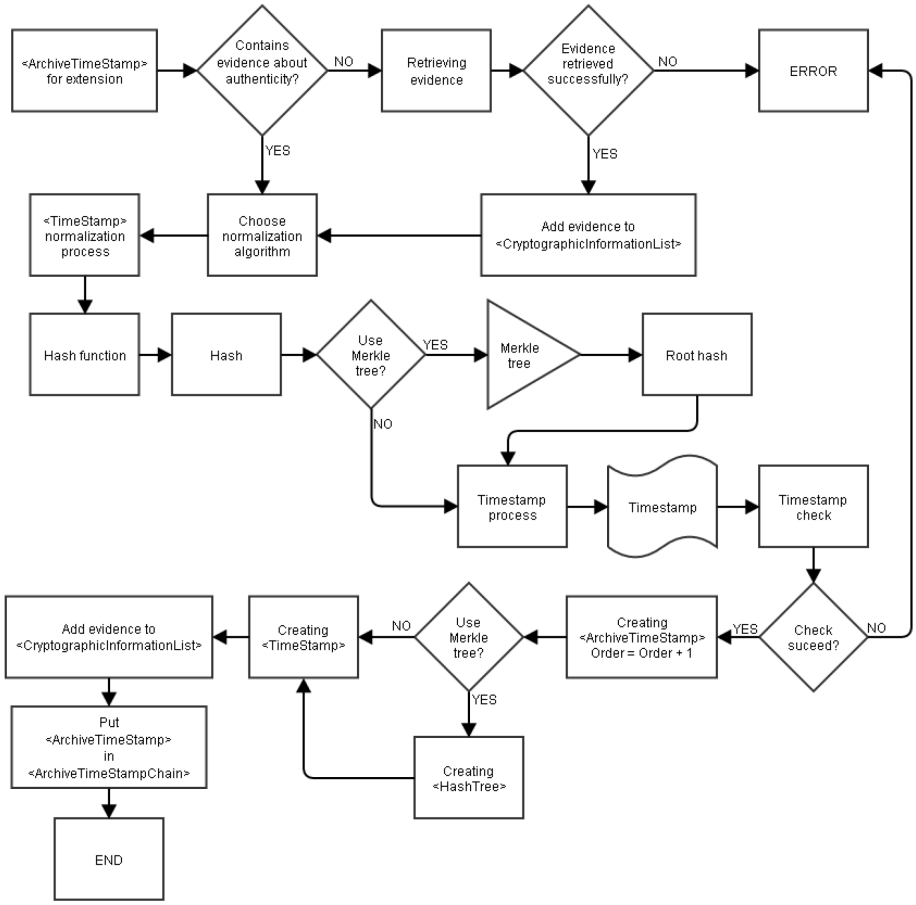


Image 21: Simple archive timestamp renewal

### 3.6.4.6 Complex renewal

Complex renewal takes place when a weak hashing algorithm is replaced.
It involves the recalculation of hashes using a new, strong algorithm. In this case, all authenticity proofs made using the weak algorithm must be complexly renewed for all AIPs created in the long-term storage process.

 

The authenticity proof renewal process is described below:

1. A new, strong hashing algorithm is selected.
2. A new algorithm is selected for conversion to a canonical form.
3. The new hashing algorithm is used to calculate the hash of the processed AIP.
4. The last archive timestamp in the current timestamp chain is checked in the accompanying ERS to see if it contains all required cryptographic elements for proving authenticity. If it does not contain these elements, they are obtained and added to the accompanying list.
5. If proofs cannot be obtained, the process cannot be continued.
6. The ArchiveTimeStampSequence tag and its content are taken from the accompanying ERS and are converted to a canonical form using the new conversion algorithm.
7. The new hashing algorithm is then used to calculate the hash for the canonical content.
8. The hashes obtained from the AIP and the canonical content are combined (their binary values are sorted in ascending order and they are joined). The new hashing algorhythm is used to calculate the hash, which is then added to the Merkle tree.
9. Steps 3 to 7 are repeated for all AIPs (and their associated ERSs) that need to be complexly renewed.
10. The timestamp is verified. If verification fails, the process cannot be continued.
11. A new ArchiveTimeStampChain tag with the Order attribute is created in the ArchiveTimeStampSequence tag. Its value is one greater than the value of the last ArchiveTimeStampChain tag.
12. The new hashing algorithm and the new algorithm for conversion to a canonical form are recorded in the new ArchiveTimeStampChain tag (the DigestMethod and CanonicalizationMethod tags). A new ArchiveTimeStamp tag with an Order attribute value of 1 is created. A new archive timestamp chain is now being built.
13. The reduced Merkle tree for the combined hash of the AIP and ERS is included in the ArchiveTimeStamp tag. A TimeStamp tag is created. The verified timestamp is added to the tag, along with a list of cryptographic elements used in the verification process.
14. Steps 10 to 12 are repeated for all ERSs included in the complex authenticity proof renewal process.

 

Complex renewal with a Merkle tree is shown in the illustration below.

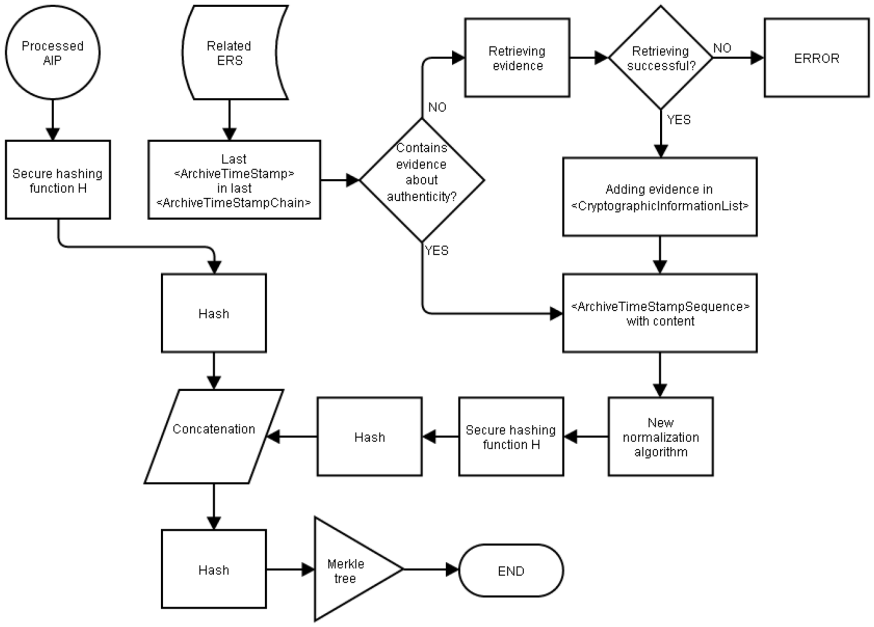


Image 22: Processing an AIP and its associated ERS

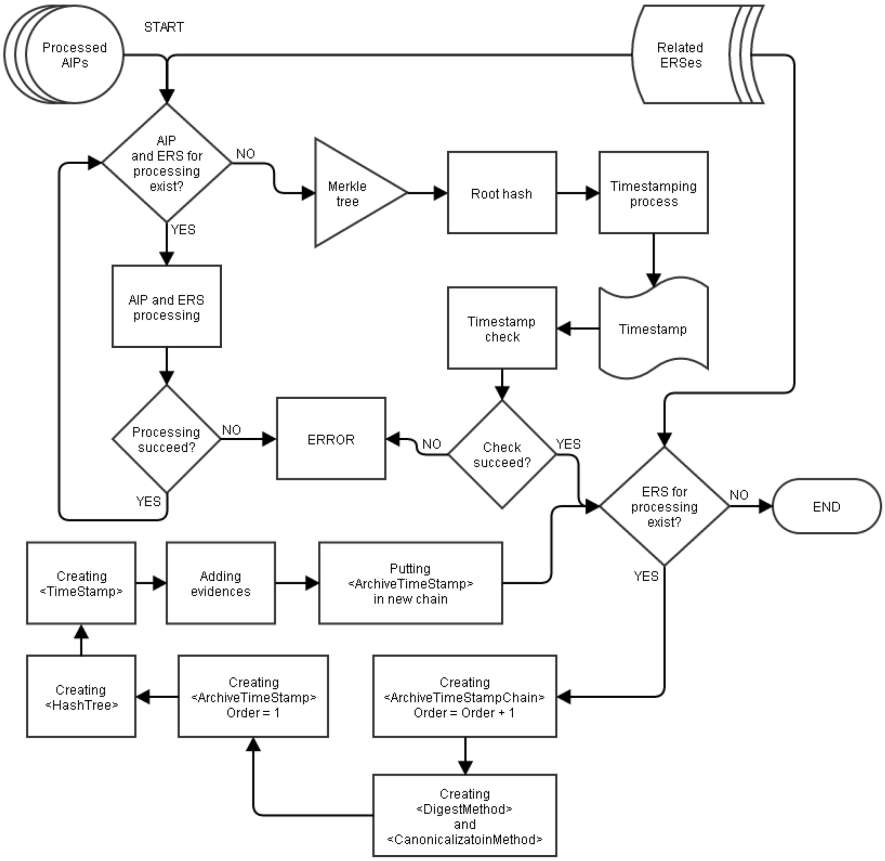


Image 23: Complex renewal with a Merkle tree

#### 3.6.4.7 Verification

ERS verification is the process in which the authenticity of all archive timestamp chains is checked. This proves the authenticity of the AIP secured by the ERS.

Verification can essentially be broken down into the following:

- Verification of the starting timestamp chain.
- Verification of the other timestamp chains.

##### 3.6.4.7.1 Verification of the starting timestamp chain

Verification of the starting timestamp chain follows these steps:

- The algorithm of the hashing function and the algorhythm for canonicalization are obtained from the chain (the DigestMethod and CanonicalizationMethod tags).
  If the DigestMethod tag does not exist, it is obtained from the data from the first archive timestamp chain (TimeStampToken).
- Once obtained, the hashing algorhythm is used to calculate the hash of the associated processed AIP.
- If the first archive timestamp contains a reduced Merkle tree, the calculated hash of the AIP must be located in the first Sequence tag, in the HashTree tag. A root hash is calculated from the reduced tree, and this hash must match the hash in the timestamp.
- If there is no reduced Merkle tree, the calculated hash of the AIP must match the hash of the timestamp.
- The digital certificate of the timestamp is checked at the time it was created (along with its associated cryptographic objects).
- The same verification is performed for all subsequent timestamps in the chain.
  Instead of an AIP hash, the previous archive timestamp is taken and canonicalized.
  The canonicalized data is used to calculate a hash and these hash values are checked in the reduced Merkle tree or in the timestamp (see the previous step).

##### 3.6.4.7.2 Verification of the other timestamp chains

The verification of chains that are not the first chain follows these steps:

- The algorithm of the hashing function and the algorithm for canonicalization are obtained from the chain (the DigestMethod and CanonicalizationMethod tags).
  If the DigestMethod tag does not exist, it is obtained from the data from the first archive timestamp chain (TimeStampToken).

 

- Once obtained, the hashing algorithm is used to calculate the hash of the associated processed AIP.
- The current chain and all subsequent chains are removed from the ArchiveTimeStampSequence tag. As a result, only previous chains remain in the ArchiveTimeStampSequence tag. These chains are the basis for canonicalization.
  As described in the section about complex renewal, the entire ArchiveTimeStampSequence is canonicalized and a hashing function is used to calculate the hash of the canonicalized data.
- The hashes of the AIP and ArchiveTimeStampSequence tag are combined (sorted in ascending order and joined) and their joint hash is calculated.
- If the first archive timestamp contains a reduced Merkle tree, the joint hash must be located in the first Sequence tag, in the HashTree tag. A root hash is calculated from the reduced tree, and it must match the hash in the timestamp.
- If there is no reduced Merkle tree, the joint hash must match the hash in the timestamp.
- The verification of the other timestamp chains follows the steps described in the section on verifying the starting timestamp chain.

Once all chains have been successfully verified using the steps described above, the content of the AIP can be said to have remained unchanged over time (from its creation to the moment of verification). If the data in the AIP match the data in the tag, the authenticity of the archived entity is guaranteed. If the verification fails at any point, the authenticity of the archived entity cannot be guaranteed.

### 3.6.5 AIP

The Archival Information Package or AIP is a XML file with summary of the metadata and content of an entity that is being secured through the generation of authenticity proof elements.
The AIP is needed to group the metadata and content of the entity into an Archival Information Package that represents the content for long-term data storage.
*(see chapter 3.6.1 Conditions).*
An AIP is generated in the following cases:

- If the properties of the entity correspond to at least one rule.
- When a closed entity has at least one metadata or content element marked for inclusion in an AIP.

For more information *see chapter 3.6.1 Conditions* and *chapter 3.3.4 Templates*.

 

The AIP is saved in the IMiS®/ARChive Server database. It can be accessed via an IMiS®/Client, which can obtain the AIP (together with its ERS) when needed, even during exporting.

The AIP record consists of the following sections:

- Header
- Metadata
- Digital certificates
- Information about revoked digital certificates.

#### 3.6.5.1 Header

All data necessary for the correct processing and interpretation of the AIP are located in the Header section. The header is a required part of the AIP.
The Header tag contains the elements listed in the table below.

| XML element name | XML element type | Required |
|---|---|---|
| Version | Attribute | YES |
| CanonicalizationMethod | Tag | YES |

Table 14: XML elements of the Header tag

"Version" attribute

The Version attribute is an unsigned 32-bit number that represents the version of the AIP.
The version prescribes the method that should be used to process and interpret the AIP when verifying the authenticity of the entity.

The values of the Version attribute are described in the table below.

| Attribute value | AIP interpretation |
|---|---|
| 1 | The AIP of version 1 is treated as a whole.<br>On the basis of the value of the CanonicalizationMethod tag a prescribed canonicalization method is selected. It will be used to convert the entire XML into a canonical form. This form is the basis for processing (calculating a hash, etc.) and verifying the authenticity of the contents and metadata of an entity. |

Table 15: The AIP interpretation depends on the value of the Version attribute

 

"CanonicalizationMethod" tag

This tag contains the Algorithm attribute. Its value is the URI or Uniform Resource Identifier that prescribes the algorithm for converting the AIP to a canonical form.

The W3C consortium prescribes the following URI values:

- *http://www.w3.org/TR/2001/REC-xml-c14n-20010315* (algorithm version 1.0).
- *http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718* (exclusive algorithm version 1.0).
- *http://www.w3.org/TR/2008/REC-xml-c14n11-20080502* (algorithm version 1.1).

*Example: The XML shows a version 1 AIP that uses a version 1.0 algorithm for conversions to a canonical form.*

```
<aip:Header version="1">
    <ds:CanonicalizationMethod algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
</aip:Header>
```

### 3.6.5.2 Metadata

The Metadata section contains the metadata of the entity defined in the templates for inclusion in an AIP *(see chapter 3.2.3 Links to templates)*, regardless of whether these metadata contain any values.

This section consists of a sequence of Attribute tags that contain the elements listed in the table below.

| XML element name | XML element type | Required |
|---|---|---|
| Id | Attribute | YES |
| Type | Attribute | YES |
| Value | Tag | NO |

Table 16: XML elements of the Attribute tag

"Id" attribute

The value of the Id attribute is the name of the metadata.

"Type" attribute

The value of the Type attribute represents the metadata type *(see chapter 3.2.1 Attribute Types)*

"Value" tag

 

The value of the Value tag represents the value or values (in the case of attributes with multiple values) of the attribute.

Content can be simple (only the value of the attribute or the finger print of the value can be present) or complex (in addition to the content or the finger print, data on digital signatures of the content are present). When a finger print of the value is present, the tag also includes the »Digest« tag with elements described in the following table.

| XML element name | XML element type | Required |
|---|---|---|
| DigestMethod | Tag | YES |
| DigestValue | Tag | YES |

Tabela 1: XML elements of the »Digest« tag

»DigestMethod« tag

The tag includes the attribute »Algorithm« which is mandatory and whose value is URI.

This prescribes an algorithm for calculating the finger print of the content.

Supported algorithms are listed in the table below which represents the supported algorithms for finger print calculation and corresponding URI, defined at W3C Consortium (*http://www.w3.org/TR/2013/NOTE-xmlsec-algorithms-20130124*) and in the specification RFC 4051 (*http://www.ietf.org/rfc/rfc4051.txt*).

| Algorithm | URI |
|---|---|
| MD5 | *http://www.w3.org/2001/04/xmldsig-more#md5* |
| SHA1 | *http://www.w3.org/2000/09/xmldsig#sha1* |
| SHA224 | *http://www.w3.org/2001/04/xmldsig-more#sha224* |
| SHA256 | *http://www.w3.org/2001/04/xmlenc#sha256* |
| SHA384 | *http://www.w3.org/2001/04/xmldsig-more#sha384* |
| SHA512 | *http://www.w3.org/2001/04/xmlenc#sha512* |

Tabela 2: Algorithms and corresponding URI's

»DigestValue« tag

The label represents Base64 coded value of a digital finger print created with an algorithm that is recorded in the attribute »Algorithm« in the »DigestMethod« tag.

 

*Example: The simple value represents the metadata of an entity where the name and address of the author are listed.*

- *The Author metadata is of the String50 type, and the Client Address1 metadata and Client Address2 are of the String100, MultiValue type.*
- *The Client Address2 does not contain a value but is still included in the AIP.*

```
<aip:Attribute Id="Author" Type="22">
    <aip:Value>John Smith</aip:Value>
</aip:Attribute>
<aip:Attribute Id="Client Address1" Type="23">
    <aip:Value>Broadway 23</aip:Value>
    <aip:Value>1515 Broadway, New York </aip:Value>
</aip:Attribute>
<aip:Attribute Id="Client Address2" Type="23"/>
```

*Example: simple content of the entity's metadata, where digital finger print of the content is used for value.*

```
<aip:Attribute Id="sys:Content" Type="42">
    <aip:Value>
    <aip:Digest>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>xcwyBBmUzlLqoHNSt+09lGdmCiw=</ds:DigestValue>
    </aip:Digest>
    </aip:Value>
</aip:Attribute>
```

Complex content also includes data on digital signatures of the content. A digital signature is included in the tag »Signature« and coded in Base64. A tag includes elements described in the table below.

| XML element name | XML element type | Required |
|---|---|---|
| Version | Attribute | YES |
| Type | Attribute | YES |

Tabela 3: XML elements of the »Signature« tag

»Value« tag

The value of the tag represents a version of the digital signature.

 

»Type« attribute

The value of the attribute represents a type of the digital signature. Currently supported format is »XMLDSIG«.

The following example shows a complex content, where in addition to a digital finger print two digital signatures of the content are also included.

```
<aip:Attribute Id="sys:Content" Type="42">
    <aip:Value>
    <aip:Signature Version="1" Type="XMLDSIG">ajM5dHogIGlqZ...</aip:Signature>
    <aip:Signature Version="1" Type="XMLDSIG">Ym52ODMgI...</aip:Signature>
    <aip:Digest>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>xQmFgsuudE4gOkT3Hg/QB65caiQ=</ds:DigestValue>
    </aip:Digest>
    </aip:Value>
</aip:Attribute>
```

### 3.6.5.3 Digital certificates

Entire chains of digital certificates that belong to the electronic signatures and that are Base64 encoded.

This section consists of the »Certificate« tag, which includes a mandatory attribute »Type« that represents the type of digital certificate. Currently supported format is »X509DER«.

The following record shows an example of two digital certificates.

```
<aip:Certificate Type="X509DER">bmlzOHYg...</aip:Certificate>
<aip:Certificate Type="X509DER">Ym4zdmJk...</aip:Certificate>
```

### 3.6.5.4 Information about revoked digital certificates

Both the chronological validity of digital certificates and information about revocations are of key importance.

An issuer may recall a digital certificate for a number of reasons (they no longer trust the confidentiality of the private key, they no longer trust the issuer of the digital certificate, etc.)

A revoked digital certificate is no longer valid (trustworthy). All digital signatures in timestamps created along with the certificate also cease to be valid.

The CRL and OCSP types of information about revocations are supported in the AIP; both types are Base64 encoded.

 

This section consists of the RevocationData tag that contains the elements listed in the table below.

| XML element name | XML element type | Required |
|---|---|---|
| Id | Attribute | YES |
| Type | Attribute | YES |

Table 17: XML elements of the RevocationData tag

"Id" attribute

The value of the attribute is a unique identifier.

"Type" attribute

The value of the attribute represents the type of information about revoked digital certificates. Supported values are shown in the table below.

| Value | Description |
|---|---|
| CRLDER | This information is a list of revoked digital certificates in binary form. |
| OSCP | This information results from continuously checking the status of a digital certificate in binary form. |

Table 18: Supported types of information about the revocation of a digital certificate

*Example: An XML file for both types of supported information.*

```
<aip:RevocationData id="1" type="CRLDER">MIISKTCCERECAQEwDQ...</aip:RevocationData>
<aip:RevocationData id="2" type="OCSP">MIIDBjCB7wI...</aip:RevocationData>
```

### 3.6.6 Time stamping

Timestamps are the proof of the existence of the content at the time listed in the stamp. Time stamping services are provided by trustworthy providers. Providers must fulfill strict security regulations, otherwise there would be reason to doubt the integrity of the timestamps they provide.

 

IMiS®/ARChive Server uses the plug-in concept to obtain timestamps. IMiS®/ARChive Server offers a plug-in for every timestamp provider that provides timestamps using heterogenous methods/interfaces. The plug-in can communicate with the timestamp provider and can include its timestamps in processes for ensuring the authenticity of records stored on the server.

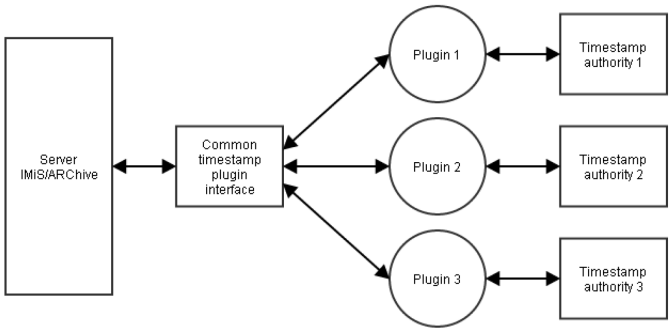


Image 24: The plug-in concept

### 3.6.6.1 Obtaining a timestamp

The process for obtaining a timestamp has the following steps:

1. The server calculates the hash of the data that are the subject of the time stamping.
2. The hash is then sent through the common interface to the plug-in for time stamping.
3. If the timestamp provider uses a nonce or arbitrary number, the plug-in uses a random number generator to create an arbitrary value which is sent to the provider along with the hash. Nonces or arbitrary numbers prevent replay attacks (*http://en.wikipedia.org/wiki/Replay_attack*).
4. The timestamp provider adds a time component to the data it receives, and signs all the components with a private key.
5. If the plug-in used a nounce or arbitrary number, this number is checked to see if it has the same value as the timestamp. If this number is not found, the timestamp is denied, as it is considered invalid.

### 3.6.6.2 Verifying a timestamp

Timestamp verification is performed by the server. The verification process follows these steps:

- The validity of the digital certificate used to create the timestamp is checked.
- Information about the revocation of any certificate in the chain, including the one that created the timestamp, is checked.

 

- The mTimestamp parameter is checked in the extensions of the digital certificate that created the timestamp *(see chapter 3.6.3.7 Extended key usage)*.
- The validity of the digital signature of the timestamp is checked.

If any of these checks is not successful, the timestamp is treated as invalid and is denied by the server. In this case, the process for obtaining a timestamp must be repeated.

### 3.6.6.3 Example

The example below shows a timestamp in XML format. The server uses the XMLSec open source library to check the timestamp in the XML format *(http://www.aleksey.com/xmlsec)*.


```
<dsig:Signature xmlns:dsig=http://www.w3.org/2000/09/xmldsig# id="TimeStampToken">
<dsig:SignedInfo>
<dsig:CanonicalizationMethod algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
<dsig:SignatureMethod algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<dsig:Reference URI="#TimeStampInfo-13ED106F54C2C33ED420000000000007B81">
<dsig:DigestMethod algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<dsig:DigestValue>LaeChaxwlaM8e9WZlRD0QjxzFrw=
</dsig:DigestValue>
</dsig:Reference>
<dsig:Reference URI="#TimeStampAuthority">
<dsig:DigestMethod algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<dsig:DigestValue>j8bwhFukHoD6jcjmzgEZtXDF/ko=
</dsig:DigestValue>
</dsig:Reference>
</dsig:SignedInfo>
<dsig:SignatureValue>
sxbZkdzdWPcXHP06k1WhZzglyTtYLXUqaOzPaT/7VkwJ3haur5yiL
qDCO1zRogaPojVC06Ee545/VrdP1JmnzcwFXAW3UU+q6VrDDHLlyD
z4uW9hXxEy31YNkQmJ7BOicHNY9m2TOlk8tjCSec6s5UJxhJP49tY
u8wE7gMSgWpLlnMeAZCE/DVOPlqesVTUYzaLSbEqELpL5qFkvCmNC
TBjnaNuyKe/YhQWbvZ0clvHePqyADNwX+IPOsA0S8NezpZHYriBO8
B+cAwglep/gZb1h8zDlqejHS8ibnFmvblk3Z0lbG/Y1SK36yk+Fu5
ya12KHlFOACzx/im3GE8vlWQ==
</dsig:SignatureValue>
<dsig:KeyInfo id="TimeStampAuthority">
<dsig:X509Data>
<dsig:X509Certificate>
MIIFYDCCBEigAwIBAgIEQLMGwDANBgkqhkiG9w0BAQUFADA+MQswC
QYDVQQGEwJzaTEbMBkGA1UEChMSc3RhdGUtaW5zdGl0dXRpb25zMR
IwEAYDVQQLEwlzaXRlc3QtY2EwHhcNMTIwNTIxMDc0ODM4WhcNMTc
wNTIxMjExMDU2WjBvMQswCQYDVQQGEwJzaTEbMBkGA1UEChMSc3Rh
dGUtaW5zdGl0dXRpb25zMRIwEAYDVQQLEwlTSVRFU1QtQ0ExLzAUB
gNVBAMTDXRzYS10ZXN0LTIwMTIwFwYDVQQFExAxMTExMTExMTExMT
ExMTExMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvsX
FScylvL5dbkj/df82Ego08QH41xuz5TcKoBHCME+9fKcNiGVivJll
```

sZOihB4jta1JABKCEA4yJDEphNV9IK5MIYaqYW/3trijvCSZefGWr
W0kHt3gclDLpR2SbGDEACxpledlA2MZR1bnSDekgCOGbPiiz/jk4Q
PDGaGZ00/Fl5WfazvKqwnvHySsoSiUk7uUH8XgAv3Uv/ghYPjSfas
xXSvy/SKJAusKOyVFXiiT+SmaX+p60KroDkxhuKM9WzoehHcCPLzy
2JUPerr02+AX4zBytPsb+AVS9XjI3KQIFpovSIph510H9hTused1g
b7gizt/leY3T3Nf4BgIxwIDAQABo4ICMzCCAi8wDgYDVR0PAQH/BA
QDAgeAMBYGA1UdJQEB/wQMMAoGCCsGAQUFBwMIMEAGA1UdIAQ5MDc
wNQYKKwYBBAGvWQEFAzAnMCUGCCsGAQUFBwIBFhlodHRwOi8vd3d3
LmNhLmdvdi5zaS9jcHMvMBoGA1UdEQQTMBGBD3RzYS10ZXN0QGdvd
i5zaTAbBgNVHQkEFDASMBAGCSqGSIb2fQdEHTEDAgEKMIH2BgNVHR
8Ege4wgeswVaBToFGkTzBNMQswCQYDVQQGEwJzaTEbMBkGA1UEChM
Sc3RhdGUtaW5zdGl0dXRpb25zMRIwEAYDVQQLEwlzaXRlc3QtY2Ex
DTALBgNVBAMTBENSTDMwgZGggY6ggYuGWGxkYXA6Ly94NTAwLmdvd
i5zaS9vdT1zaXRlc3QtY2Esbz1zdGF0ZS1pbnN0aXR1dGlvbnMsYz
1zaT9jZXJ0aWZpY2F0ZVJldm9jYXRpb25MaXN0P2Jhc2WGL2h0dHA
6Ly93d3cuc2lnZW4tY2Euc2kvY3JsL3NpdGVzdC9zaXRlc3QtY2Eu
Y3JsMCsGA1UdEAQkMCKADzIwMTIwNTIxMDc0ODM4WoEPMjAxNTA1M
jEyMjE0NDlaMB8GA1UdIwQYMBaAFFRJB0aHzx2JncqucqeooKBpty
HnMB0GA1UdDgQWBBQS8wflsAtvqW4RKcSTY5n4Mbg/dTAJBgNVHRM
EAjAAMBkGCSqGSIb2fQdBAAQMMAobBFY3LjEDAgSwMA0GCSqGSIb3
DQEBBQUAA4IBAQBh2Vgmo+MrbwqVcSvMln7aA9HLjw+HlM5KbXXvM
/XEmFWhVrWFhbqX1f+OoizSfDv4sErCPb3zzeSG2mnEREzk9gqzbt
NlqfvzdZ7Xd7bGUEzwHEcVE8DSW2bJGeSkhyX2AWvt4eHyqwgXply
MzFOVSWteivgwgvMtUZjSbeClpELUblGWzwyxW15SHufOdJdtgcnn
r2hCLtkYJ9ky4T0m5gvvyOxmQi+o3rLvPa5yLeHYa//KzPo+H8CPf
UGqRFTPVSykaIm6evTg6CDQvG9jfii1PbaeKQRm0xLQPpVNYFA66e
2DwVv+9UdKzcoBfujvZRtEuPlnphHBlzxEkFU3
</dsig:X509Certificate>
</dsig:X509Data>
</dsig:KeyInfo>
<dsig:Object id="TimeStampInfo-13ED106F54C2C33ED420000000000007B81">
<ts:TimeStampInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:ts="http://www.entrust.com/schemas/timestamp-protocol-20020207">
<ts:Policy id="http://www.si-tsa.si/dokumenti/SI-TSA-politika-za-casovni-zig-1.pdf"/>
<ts:Digest>
<ds:DigestMethod algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>93WQD+wSgMA5KCzcmjYe55NBKEc=
</ds:DigestValue>
</ts:Digest>
<ts:SerialNumber>1084876374609840033695664166241473103450 89
</ts:SerialNumber>
<ts:CreationTime>2014-03-31T15:00:00.089Z
</ts:CreationTime>
<ts:Nonce>10600496071266535864
</ts:Nonce>
</ts:TimeStampInfo>
</dsig:Object>
</dsig:Signature>

“dsig:SignedInfo” tag

This tag contains information about what part of the XML is the subject of the digital signature and methods for canonicalizing the XML when verifying a timestamp:

- dsig:CanonicalizationMethod: The method for canonicalizing the XML.
- dsig:SignatureMethod: The algorithm used to create the digital signature.
- dsig:Reference URI="#TimeStampInfo-13ED106F54C2C33ED420000000000007B81": A reference to the dsig:Object that is the subject of the content of the digital signature. It also contains the algorithm and hash of that entity.
- dsig:Reference URI="#TimeStampAuthority": A reference to the dsig:KeyInfo tag, which is also the subject of the digital signature content. It contains the algorithm and hash of that entity.

“dsig:SignatureValue” tag

This tag contains the value of the digital signature.

“dsig:KeyInfo” tag

This label contains information about the digital certificate used to create the timestamp:

- dsig:X509Data – This tag contains the dsig:X509Certificate tag, which contains the digital certificate used to create the digital signature.

“dsig:Object” tag

This tag contains the following data:

- ts:TimeStampInfo: Information about the protocol.
- ts:Policy: Information about the policy of the timestamp provider.
- ts:Digest: The hash (and the URI algorhythm used to create it) the timestamp provider received for time stamping.
- ts:SerialNumber: The serial number.
- ts:CreationTime: The date and time of the creation of the timestamp.
- ts:Nonce: A nonce or arbitrary number used in the process of creating the timestamp request.

 

### 3.6.7 Rules

The rules specify the conditions under which the processes for generating and renewing authenticity proofs are carried out and the content that forms the subject of these processes. Generating and reviewing proofs is performed in packages using a Merkle tree unless there is only one AIP or archival timestamp in the package. Packages are defined as a minimum and maximum number of archival information packages that are time-stamped with one timestamp. An example of package configuration is the following XML record:

```
<Settings>
    <MinBatchSize>1</MinBatchSize>
    <MaxBatchSize>300</MaxBatchSize>
</Settings>
```

»MinBatchSize«

Includes a minimum number of archival information packages that are time-stamped with one timestamp. The value can't be lower than 1 or bigger than value »MaxBatchSize«.

»MaxBatchSize«

Includes a maximum number of archival information packages that are time-stamped with one timestamp. The value can't be lower than »MinBatchSize« or bigger than 1.000.000.

#### 3.6.7.1 Rules for generating authenticity proofs

Rules for generating authenticity proofs are used when checking whether an entity meets the conditions to ensure authenticity. If the properties of the entity correspond to at least one rule, proof will be generated for this entity as defined by the rule.

The example of the rule is shown in the following XML record:

```
<Rule>
    <Id>307a8c60-390c-470d-9692-a43311bd51ef</Id>
    <Enabled>true</Enabled>
    <Type>Explicit</Type>
    <IncludeChildren>true</IncludeChildren>
    <Scope type="ClassificationCode">C=57</Scope>
    <Expression>[sys:Status] = "2"</Expression>
    <TemplateFilter></ TemplateFilter>
    <TimeStampProviderId>a48d9c39-456a-470c-9c8b-e54bb91fc1dc</TimeStampProviderId>
</Rule>
```

 

»Id« attribute
The attribute includes a unique identifier rule.

»Enabled« tag
The value of the tag can be »True« or »False« and determines whether a rule is enabled or not.

»Type« tag
The value of the tag can be »Explicit« or »Implicit«. Explicit rules are checked when performing operations with entities (save, change status...), while in the process of time-stamping all rules are checked.

»IncludeChildren« tag
The value of the tag can be »True« or »False«, and it determines the validity of rules for subentities.

»Scope« tag
The value of the tag includes an identifier of the entity under which the rule is taken into account when checking the properties of the entity. If the entity is not included, then such rule does not apply to it.

»Expression« tag
The value includes a search string that is checked with the properties of the entity.
If a search string does not correspond to the values in the entity, then such rule does not apply to the entity.

»TemplateFilter« tag
The value of the tag includes the names of the template with which the entities must be created in order to correspond to the rule. The value is used as an additional filter for value »Expression«.
If the value »TemplateFilter« is an empty string, then the value is not taken into account when checking the rule with the value of the entity.

»TimeStampProviderId« tag
The value of the tag represents a unique identifier of the timestamp provider with which the entities that correspond to the rule will be time-stamped.

*Example: The rule in our cause applies to entities in a class with a classification code »57« (applies to the class as well) and have a value »sys:Status« equal to 2 (closed entity) or are subentities of a class or a closed case (i.e. documents in a closed case).*

 

#### 3.6.7.2 Rules for renewing proofs

Rules for renewing proofs are used for simple and complex renewals.
An example is shown in the following XML record:

```
<Rule>
    <Id>6062074f-6d05-4f24-8592-866623ade8bf</Id>
    <Enabled>true</Enabled>
    <Digest>SHA1</Digest>
    <CertificateExpirationTreshold>90d</CertificateExpirationTreshold>
    <TimeStampProviderId>bac87d4d-fd1b-4065-bb36-5d03eeb25b6e</TimeStampProviderId>
</Rule>
```

»Id« attribute
The attribute includes a unique identifier rule.

»Enabled« tag
The value of the tag can be »True« or »False«, and determines whether the rule is enabled or not.

»Digest« tag
The value of the tag represents a type of hash function that will be used with time-stamping.

»CertificateExpirationTreshold« tag
The value represents a time frame in which all timestamps with the same type of hash function, whose digital certificates expire in the given time frame, will be captured.

»TimeStampProviderId« tag
The value represents a unique identifier of the trustworthy timestamp provider with which the proofs will be renewed before they expire.

Therefore, the rule in our case applies for all proofs generated with a hash function SHA1, whose digital certificates expire within 90 days.

## 3.7 Review process

Review process is a monitored and planned process for implementation of transfer, destruction or permanent retention of electronic documentation. Each class, folder or document (classified directly under class) must have at least one determined retention period (*see chapter 3.3.9.2 Managing retention policy connections*). It specifies how much time an entity must be archived in the archive system.

 

A retention period contains a time frame and a default action that will be implemented in the review process. The action can be changed by users with rights (members of the committee) that implement review.
The review process is divided into three phases:

- Preparation
- Decision-making
- Implementation.

In the preparation and implementation phase of the review process, errors can occur for various reasons. In the case of an error, the process is automatically cancelled.
The following changes are implemented:

- The status is changed to Closed.
- The state is changed to Failed. The »sys:ret:rev:State« attribute contains the value of the state.
- The reason for cancellation is recorded in the »sys:ret:rev:Message« attribute,

The process in which an error occurred cannot be prepared or implemented again.
Editing is also not allowed. It should be emphasized that in the preparation and implementation phase of the review process, actions on entities are implemented in the name of the user that launched the preparation or implementation. User rights and the security level on the entities affect the process of preparation and implementation. A course diagram of the review process in preparation or implementation phase is shown below.

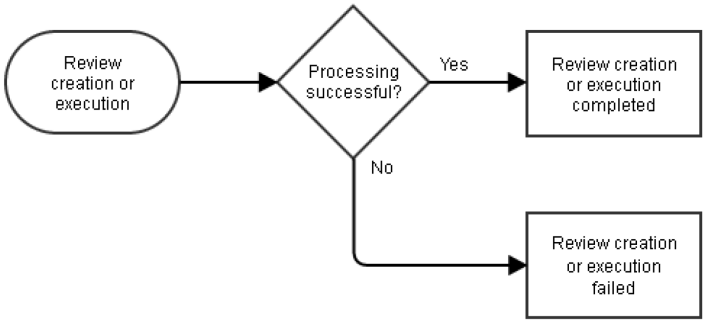


Image 25: Course diagram of the review process

 

### 3.7.1 Preparation phase

The first phase of the review process is preparation. A user with rights creates an entity that will be later on included in the review process.

Besides the mandatory attributes, the user must enter the value of one of the following attributes:

- »sys:ret:rev:Query«: the value represents a search string for entities *(see chapter 3.5.2 Search syntax rules)*, that will be included in the review process.
  The use of a search string in the preparation phase of the review process is intended for ad-hoc managing of entities without the retention policies.
- »sys:ret:rev:Schedules«: values represent unique identifiers of retention policies that will be taken into consideration in the preparation phase of the review process.

If both attributes have determined values, the preparation phase of the review process is cancelled as this is an invalid state.

The »sys:ret:rev:Scope« attribute plays an important role in the review process. The value of the attribute is a classification code of the entity under which a search of all entities that match the conditions will be implemented. If the value of the attribute is not set, the search will occur in the entire archive.

The preparation phase of the review process is divided into:

- Preparation based on the search string
- Preparation based on retention periods.

In the preparation phase based on a search string, a search of entities that meet the conditions in the search string is implemented. The result is a collection of entities that are subject to filtration. Filtration removes all entities from the collection that do not belong in the review process *(see chapter 3.7.4 The filtration process)*. Then XML documents are created from the filtrated collection that contain information on entities and allowed actions that users with rights (members of the committee) manage in the decision-making phase of the reviw process.

 

The preparation phase is similar with retention periods. A search string is taken from the trigger (value of the »sys:ret:pol:Trigger« attribute) from each retention policy that is a part of the review process, and an entity search is implemented. When the search is completed in all retention periods, the collections of all searches form a union that represents a collection of entities from all retention periods. This collection undergoes a filtration process where it is additionally checked whether entities have at least one effective retention period that is part of the review process. If this condition is not met, the entity does not belong to the current review process. Finally, XML documents are created from the filtrated collection and they contain information on entities and allowed actions that a user can change in the decision-making phase.

If an error occurs in the preparation phase (e.g. an invalid search string, a deleted retention policy, etc.), the process is automatically cancelled. For a new preparation, the entire phase of preparation must be repeated.

The preparation phase of the review process is shown below.

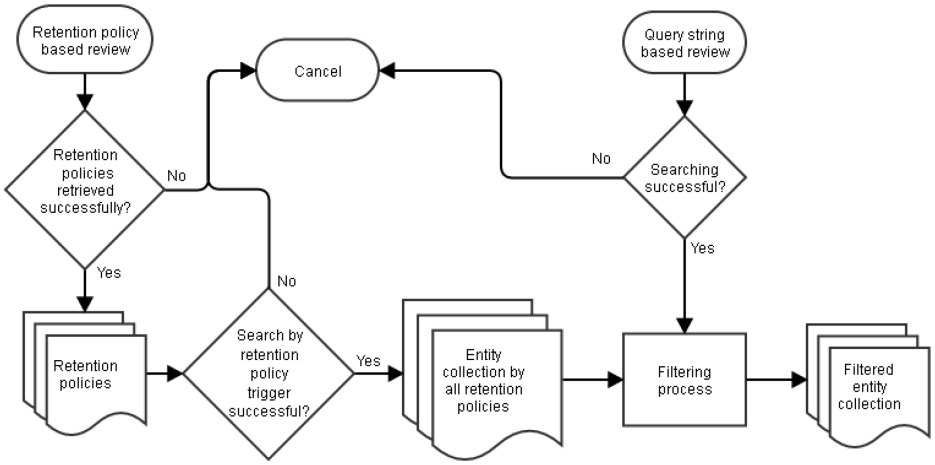


Image 26: Preparation phase of the review process

After a successfully implemented preparation process, XML documents that represent the content of the process are created.

There are two types of XML documents:

- Read-Only documents (in the »sys:ret:rev:Lists« attribute).
- Documents that members of the committee change in the decision-making phase (in the »sys:Content« attribute).

 

A Read-Only document contains information on the entity and a set of actions that can be implemented on the entity. The set of actions depends on several factors that are described in detail *in the chapter 3.7.4 The filtration process*. Documents that the members of the committee can change in the decision-making phase contain information on the entity, the default action and reason.

### 3.7.2 Decision-making phase

In the decision-making phase, the members of the committee make a decision on what happens with the entities that were part of the preparation. The members of the committee select an action that is allowed for an individual entity, add a comment, change the reason for the selected action, etc.

If the entities are transferred to another archive system, the members can add a reference to the transferred entities or confirm a successful entity transfer. If the transfer is not confirmed, entity destruction is cancelled on the server.
After the completed decision-making, the review process is implemented or cancelled.
If the process is implemented, the value of the »sys:ret:rev:Action« attribute is set to Complete. This is a sign for the server that the process has been implemented.
If the process is cancelled, the value of the attribute is set to Discard, which is a sign for the server that the review process is cancelled.

### 3.7.3 Implementation phase

Implementation of the review process is divided into:

- Cancellation of the review process.
- Implementation of the review process.

When the review process is cancelled:

- The server sets the value of the »sys:ret:rev:State« attribute to Discarded.
- The status is changed to Closed.
- A message is recorded in the »sys:ret:rev:Message« attribute that the review process has been cancelled by the user.

 

In the implementation phase of the review process, all information on entities is loaded from the XML Read-Only documents. Then verification is implemented in the following order:

- The form of XML documents must be compliant with the assigned XSD scheme.
- All entities from XML Read-Only documents must be present in the changeable XML documents and vice versa.
- Selected actions on entities must be compliant with the allowed actions for the entity.

If verification is unsuccessful due to any of the steps, the review process is implemented with an error. The entire process must be repeated.
An exception is verification of a selected action on entities. In that case, the implementation is cancelled with an error. The state of the review process is set to »InReview«. This allows the user to fix selected actions on entities and repeats the process. After a successful verification, a filtration process is implemented on the collection of entities. The filtration process is described in more detail *in chapter 3.7.4 The filtration proces*.
Here are some practical examples why a cancellation of action implementation occurs in the implementation phase of the review process.

*Example 1: In the review process, there is a folder with two contained folders besides other entities:*

*F=2015-00011*
*F=2015-00011^F=00001*
*F=2015-00011^F=00002*

*In the review process, the selected action for all three entities is »Dispose«. Members of the committee realize that a disposition hold is bound to the »F=2015-00011« folder. Implementation of the action is cancelled on all three entities as the disposition hold recursively affects all contained entities. The action is implemented on all other entities that are subject to the review process.*

*Example 2: Let's take the combination from example 1 except that now the disposition hold is not bound to the »F=2015-00011« folder. The folder contains a new folder »F=2015-00011^F=00003« that did not yet exist in the preparation phase of the review process:*

*F=2015-00011*
*F=2015-00011^F=00001*
*F=2015-00011^F=00002*
*F=2015-00011^F=00003*

 

*In this example, the »Dispose« action is implemented on contained folders »F=2015-00011^F=00001« and »F=2015-00011^F=00002«, whereas the disposition of the »F=2015-00011« folder is not possible as it contains a new folder »F=2015-00011^F=00003« that is not a subject of review process. If an error occurs during implementation on an individual entity, action implementation is cancelled on all parent entities that are part of the review process.*

*Example 3: Let's take the combination from example 1, however, two documents are in the contained folder »F=2015-00011^F=00002«:*

*F=2015-00011^F=00002*
*F=2015-00011^F=00002^D=00001*
*F=2015-00011^F=00002^D=00002*

*The user in whose name the review process is being implemented does not have the right to delete the »F=2015-00011^F=00002^D=00002« document. Consequently, the contained folder »F=2015-00011^F=00002« and its parent folder »F=2015-00011« are not disposed. However, the contained folder »F=2015-00011^F=00001« is disposed.*

*Example 4: Let's take the combination from example 1, where the action on contained folders »F=2015-00011^F=00001« and »F=2015-00011^F=00002« is set to »Permanent«. The action on the »F=2015-00011« folder is set to »Dispose«.*
*If the »Permanent« action is successfully implemented, the implementation of the »Dispose« folder is unsuccessful as the »F=2015-00011« folder contains two folders in permanent retention.*
*Consequently, the subject is not destroyed.*
*The implementation of all actions is recorded in the audit trail (if it is enabled) and separately in a report.*
*The report consists of an XML and text document.*
*Actions on entities that are part of the review process are logged in the XML document.*
*Actions on entities that are not part of the review process (documents in folders) are logged in the text document. Such entities are indirectly dependent as the parent entities define actions that will be implemented.*

### 3.7.4 The filtration process

In the filtration process, the entire collection of entities is verified whether the entities meet the conditions of the review process.

 

All entities (and their parent entities) that comply with the following conditions in the preparation phase of the review process are disposed of:

- The user in whose name the preparation is implemented has no right to see the existence of an entity.
- At least one disposition hold is present on an entity.
- An entity does not contain at least one effective retention period that is compliant with the set of retention periods that are used in the preparation phase of the review process.
  In the case of preparation of the review process on the search string, no verification occurs.
- An entity contains child entities that are subject to review process but they are not present in the collection of entities for the current preparation of the review process.

For all other entities the presence of the »sys:Significance« attribute is verified.
The following values affect the allowed actions that are available on the entity in question:

- The »Vital« or »Permanent« values on the entity: they affect the setting of the default action to »Permanent«. The user can select only the »Review« action as the attribute defines the importance of the entity. The value affects all other parent entities whose default actions are changed as a consequence.
- The »Retain« value: it represents a warning that an entity must be retained. For that reason, the default action is set to »Review«. The user can change the action. The value has no influence on the parent entities.

The following conditions affect the default action of an entity:

- If an entity has more effective retention periods, it means it is in conflict. The default action is set to »Review«. The user can choose from all the available actions.
- If an entity contains other entities that have been in other review processes and have already become permanent, the default action of the entity in question is set to »Permanent«. This also applies to all parent entities. The user can select only the »Review« action.

If the state of an entity is not compliant with any of the above conditions, the default action is set to the value that the valid retention policy for this entity contains. The user can choose from all the available actions.
The filtration process is also used in the implementation phase of the review process.

 

In that case, entities (plus all the parent entities) on which the selected actions cannot be implemented on are marked:

- At least one disposition hold is bound to the entity.
- An error occurred on the contained entities during operation of an action (e.g. the user in whose name the review process is being implemented does not have sufficient rights to implement the operation, etc.).
- The state on the server in the implementation phase of review process differs from the state in preparation phase.

  *Example: new entities that are subject to review process, but they are not present in the current process, etc.*

- The selected actions are in conflict.

  *Example: The selected action of a contained folder is »Review« and the parent folder is »Dispose«. In that case, the selected action cannot be implemented on the folder as it contains a folder that will be selected in another review process.*

### 3.7.5 XML document format

In the review process, the following types of XML documents appear:

- Read-Only documents
- Read and Write documents
- Reports.

#### 3.7.5.1 Read-Only document syntax

Read-Only documents contain a state on the server in the preparation phase of the review process. They also contain information on the entity (classification code, internal Id, address), valid actions, reason and effective retention periods that apply to this entity.

##### 3.7.5.1.1 »Review« tag

This tag represents a root element and contains elements from the table below.

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Header | Tag | Yes |
| Entity | Tag | No |

Table 19: XML elements of the “Review” tag

 

#### 3.7.5.1.2 »Header« tag

This tag contains a version of the XML document, the start date of the preparation phase of the review process, entity classification code (represents a scope of entity searches for the review process), available actions, predetermined reasons, and identifiers of all retention periods. The tag elements are described in the table below.

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Version | Attribute | Yes |
| StartDate | Attribute | Yes |
| Scope | Attribute | No |
| Action | Tag | Yes |
| Reason | Tag | Yes |
| Schedule | Tag | Yes |

Table 20: XML elements of the "Header" tag

»Version« attribute
This attribute represents a version of the XML document.

»StartDate« attribute
This attribute represents the start date and time of preparation of the review process.

»Scope« attribute
This attribute represents a classification code of an entity under which an entity search was implemented in preparation phase of the review process. If the attribute is not present, the search was conducted in the entire archive.

»Action« tags
These tags represent a set of actions that are available in the implementation phase of the review process. Entities have a required »Id« attribute.

»Reason« tags
These tags represent reasons for default actions that will be implemented on the entities.
A tag has a mandatory »Id« attribute that represents a unique identifier other tags refer to.

»Schedule« tags
These tags contain retention policies »Id« for the current review process. They are effective for individual entities and retention policies that are in the preparation phae of the review process.
A tag has a required »Id« attribute that represents a unique identifier other tags refer to.

 

#### 3.7.5.1.3 »Entity« tag

Tags contain information on the entity that is involved in the review process.

Elements of a tag are described in the table below:

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Id | Attribute | Yes |
| IdType | Attribute | Yes |
| Type | Attribute | Yes |
| Id85 | Attribute | Yes |
| Title | Attribute | Yes |
| Action | Attribute | Yes |
| Reason | Attribute | No |
| Schedule | Tag | Yes |

Table 21: XML elements of the "Entity" tag

»Id« attribute

This attribute represents an entity identifier.

»IdType« attribute

This attribute represents a type of entity identifier. The following values are valid:

- »C«: classification code is the identifier
- »E«: unique external identifier is the identifier
- »I«: internal entity identifier is the identifier.

»Type« attribute

This attribute represents a type of entity. The following values are valid:

- »C«: the entity represents class
- »F«: the entity represents folder
- »D«: the entity represents document.

»Id85« attribute

This attribute represents an internal entity »Id«.

»Title« attribute

This attribute represents the title of an entity.

 

»Action« attribute

This attribute is a 32-bit unsigned value that contains information on valid actions on an entity and the default action for an entity. The first four bit values (from right to left or from the LSB side) represent valid actions that are described in the table below. The action is allowed only if the value of the bit is »1«.

| Bit 3 – »Review« | Bit 2 – »Transfer« | Bit 1 – »Permanent« | Bit 0 – »Dispose« |
|---|---|---|---|

Table 22: Position of bites that represent allowed actions

The next 24-bites are not in use. The value of the final 4-bits represents the value of the »Id« attribute in the »Action« tag inside the »Header« tag. At the same time, it represents the default action for an entity.

»Reason« attribute

This attribute represents a reference to the »Id« attribute in the »Reason« tag in the »Header« tag, whereas the »Reason« value represents the default reason for review of entity.

»Schedule« tag

This tag contains the required »Id« attribute that contains a reference to the »Id« attribute in the »Schedule« tags in the »Header« tag: The value represents an internal retention period identifier that is effective for the entity.

*Example:*

```
<Review>
    <Header StartDate="2015-08-17T18:14:19.170+02:00" Version="1">
        <Action Id="1">Dispose</Action>
        <Action Id="2">Permanent</Action>
        <Action Id="3">Transfer</Action>
        <Action Id="4">Review</Action>
        <Reason Id="R0">Reason 1</Reason>
        <Reason Id="R1">Reason 2</Reason>
        <Schedule Id="S0">
        0d5021dd60c06fcf09fe42d6ac61a4434727e5a69d9253d5ed17ef84879e1996
        </Schedule>
        <Schedule Id="S1">
        098197cff3ea91e9feadfd2a629ec4ad0cf3d06e6025c616853a58363819c8f6
        </Schedule>
    </Header>
    <Entity Id="C=99" IdType="C" Type="C"
        Id85="jkh920856nvbghd84ikfnbj2185hwqbo58djhn378jck085jd"
        Title="Legacy object containers" Action="536870927" Reason="R1"
        <Schedule Id="S1"/>
    </Entity>
```

 

```
    <Entity Id="C=04^F=2015-00012^F=00001" IdType="C" Type="F"
        Id85="b43ghjf983jxcgfsa6r9fhnnbnd84hfcsfk93e7tb184jshfg"
        Title="Review test" Action="536870927" Reason="R1">
        <Schedule Id="S1"/>
    </Entity>
</Review>
```

*The value of the »Action« attribute for the »C=04^F=2015-00012^F=00001« entity is: 536870927.*
*The binary representation is as follows: 00100000000000000000000000001111.*
*All actions are enabled on this entity (the value of the first 4 bits is 1111). The last 4 bits produce the value 0010 (2), which is the »Id« attribute value of the »Permanent« action.*

#### 3.7.5.2 Read and Write document syntax

Sintaksa XML dokumentov, ki se urejajo v postopku odločanja je zelo podobna dokumentom samo za branje. Vsebuje nekaj dodatnih atributov, ki se uporabljajo v postopku izvajanja odbiranja in izločanja.

##### 3.7.5.2.1 »Review« tag

This tag represents the root element of a document and contains elements from the table below:

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Header | Tag | Yes |
| Entity | Tag | No |

Table 23: The “Review” tag elements

##### 3.7.5.2.2 »Header« tag

This tag contains a version of the XML document, available actions and associated reasons. The tag elements are described in the table below:

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Version | Attribute | Yes |
| Action | Tag | Yes |
| Reason | Tag | Yes |

Table 24: The “header” tag elements

 

»Version« attribute
This attribute represents a version of the XML document.

»Action« tag
These tags represent a set of available actions. They contain a required »Id« that represents a reference to the action that has been selected for an individual entity in the implementation phase of the review process.

»Reason« tag
Tags represent a set of reasons for implementation of actions on the entities in the implementation phase of the review process. They contain a mandatory »Id« attribute that represents a reference to the reason that an individual entity refers to.

#### 3.7.5.2.3 »Entity« tag

Besides attributes that refer to the entity, this tag contains a selected action and reason.
In the case of »Transfer« action, it is necessary to confirm a successful entity transfer with the attribute.
Optionally, the reference to the transferred entity and comment can be entered.

The tag elements are described in the table below.

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Id | Attribute | Yes |
| IdType | Attribute | Yes |
| Action | Attribute | Yes |
| Reason | Attribute | Yes |
| TrfSucceed | Attribute | No |
| TrfRefId | Attribute | No |
| Comment | Attribute | No |

Table 25: The "Entity" tag elements

The »Id« and »IdType« attributes represent an entity identifier and a type of identifier. They are the same as in the Read-Only document *(see chapter 3.7.5.1 Read-Only document syntax)*.

»Action« attribute
This attribute value is a reference to the selected action in the »Header« tag.

 

»Reason« attribute

This attribute value is a reference to the selected reason in the »Header« tag.

»TrfSuceed« attribute

If this attribute is present, its value indicates whether the entity transfer has been successful or not (»True« or »False« values).

»TrfRefId« attribute

This attribute contains a value that represents a reference to the transferred entity.

»Comment« attribute

This attribute contains a comment by the user that implements the decision-making.

*Example:*

```
<Review>
    <Header Version="1">
        <Action Id="A0">Dispose</Action>
        <Action Id="A1">Permanent</Action>
        <Action Id="A2">Transfer</Action>
        <Action Id="A3">Review</Action>
        <Reason Id="R0">Reason 1</Reason>
        <Reason Id="R1">Reason 2</Reason>
    </Header>
    <Entity Id="C=99" IdType="C" Action="A2" Reason="R1" TrfSucceed="true"
    TrfRefId="IMiSARC-1-192749948393985769347589569493874 85"/>
    <Entity Id="C=99^C=01" IdType="C" Action="A0" Reason="R1"/>
    <Entity Id="C=99^C=02" IdType="C" Action="A0" Reason="R1"/>
</Review>
```

*The following will occur in the process of decision-making:*

- *the »C=99^C=01« and »C=99^C=02« entities are marked for disposition (the value of the »Action« attribute is »A0, which is a reference to the »Dispose« action);*
- *the »C=99« entity has been successfully transferred (the »TrfSucceed« value has the »true« value). A reference to the transferred entity has also been entered.*

### 3.7.5.3 Report syntax

Reports contain information on entities, the selected action and information whether the selected action has been successfully implemented.

 

#### 3.7.5.3.1 »Report« tag

This tag represents the root element and contains elements from the table below.

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Header | Tag | Yes |
| Entity | Tag | No |

Table 26: The "Report" tag elements

#### 3.7.5.3.2 »Header« tag

This tag contains a version of the XML document and actions selected for individual entities. The tag elements are described in the table below.

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Version | Attribute | Yes |
| Action | Tag | Yes |

Table 27: The "Header" tag elements

»Action« tag

These tags represent a set of available actions. They contain a mandatory »Id« attribute that represents a reference to the action selected for individual entities in the implementation phase of the review process.

#### 3.7.5.3.3 »Entity« tag

Besides attributes that apply to an entity, this tag also contains a selected action and information whether the action has been successfully implemented. The tag elements are described in the table below.

| Name of XML element | Type of XML element | Required |
|---|---|---|
| Id | Attribute | Yes |
| IdType | Attribute | Yes |
| Action | Attribute | Yes |
| Error | Attribute | No |
| Message | Tag | No |

Table 28: The "Entity" tag elements

 

The »Id« and »IdType« attributes represent the identifier and identifier type of an entity, and they are the same as in the Read-Only document *(see chapter 3.7.5.1 Read-Only document syntax*).

»Action« attribute
This attribute value is a reference to the selected action in the »Header« tag.

»Error« attribute
If this attribute is present, it has the »True« value. This means that the selected action on an entity has not been successfully implemented. The reason for error is described in the »Message« tag.

»Message« attribute
If this tag is present, it contains a description of an error that occurred on the server during action implementation.

*Example:*

```
<Report>
    <Header Version="1">
        <Action Id="A0">Dispose</Action>
        <Action Id="A1">Permanent</Action>
        <Action Id="A2">Transfer</Action>
        <Action Id="A3">Review</Action>
    </Header>
    <Entity Id="C=01^C=05" IdType="C" Action="A0"/>
    <Entity Id="C=01^C=06" IdType="C" Action="A0" Error="true">
        <Message>Access denied. Insufficient rights to disposed the entity.</Message>
    </Entity>
</Report>
```

*It is evident from the report that the »C=01^C=05« entity has been successfully disposed, whereas an error occurred with the »C=01^C=06« entity as the user who implemented the review process has no rights to delete this entity.*

 

## 3.8 Directory services

Directory services enable work with the server directory. The directory contains registered users and user groups. These are an important part of the server infrastructure, as they support the following processes and operations:

- User authentication.
- Obtaining information about a registered entity in the server directory using attributes.
- Managing the Access Control List (ACL).

Authentication

Authentication is the process performed when setting up a user session.
If authentication is successful, the directory service confirms the identity of the user connecting to IMiS®/ARChive Server. A user's identity is confirmed if their user credentials match the data in the server directory.

To simplify, user credentitals are a set of information that unambiuously ensures that the authentication data used to create the user credential data match the data used to create the data in the server's database of user credentials.
In other words, a key or password uses mathematic procedures to create user credential data that tells the server that they were made using the key that the user used the last time the user credentials were changed.

Obtaining information from attributes

Directory services make it possible to obtain information about registered entities in the server directory by using the entities' attributes. This information is obtained on the basis of a unique 32-bit identifier that is assigned to every registered entity in the server directory. This identifier can be used to obtain information such as the user's account, the name and surname of the user, a description of the user and other data from the directory entity (*see chapter 3.2.1 Attribute types*).

Managing the Access Control List (ACL)

The access control list also uses a 32-bit unique identifier. Rights and roles are linked to this identifier. Directory services provide a link between an entity in the directory and its unique identifier.

 

At the same time, these services also resolve the user groups individual users belong to. This data is used by the ACL to calculate effective rights *(see chapter 3.3.5.2.6 Effective rights)*.

### 3.8.1 Types

The server directory supports two entity types:

- Users.
- User groups.

#### 3.8.1.1 Users

Users are persons who access information on the server. Users can be assigned to any number of groups. There is no limit.

*Example:*
*A user belongs to two groups, FINANCE and ACCOUNTING. The administrator can set an individual user's status as active or inactive. Inactive users will not be able to start a session with the server.*

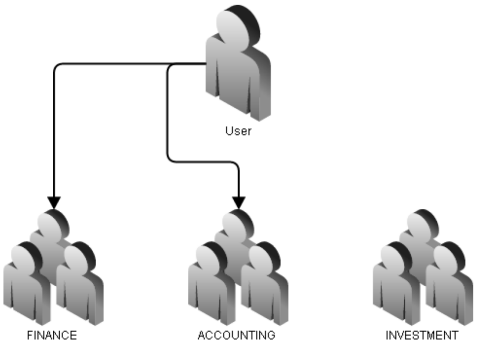


Image 27: A user and their user groups

#### 3.8.1.2 User groups

User groups are directory entities that have been assigned group rights on the server. This provides an easier way of controlling access to individual documents, classes and folders for large numbers of users. User groups can be nested as needed, and recursive nesting is also possible. If recursion is detected in a group when resolving user groups, the system skips that group, as its content has already been resolved.

*Example: The FINANCE, ACCOUNTING and INVESTMENT user groups are nested in the FINANCIAL SECTOR user group; at the same time, the FINANCIAL SECTOR user group also contains a user of its own.*

 

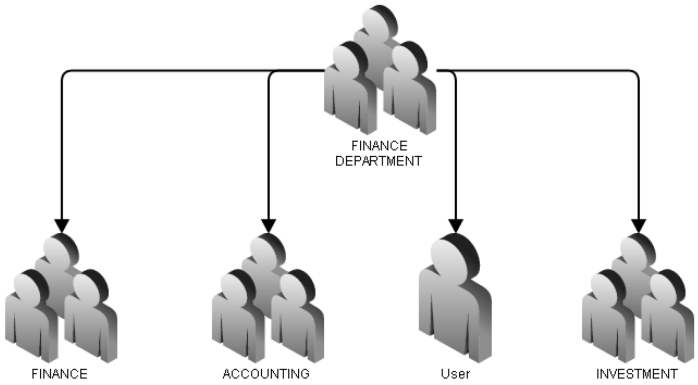


Image 28: Example of user group nesting

### 3.8.2 Entities defined by the system

The directory of IMiS®/ARChive Server has the following predefined system and normal entities:

- sys:Administrator
- sys:Administrators
- sys:CurrentUser
- sys:Server
- sys:Everyone
- Anonymous.

"sys:Administrator"

This directory entity represents the user who is the local administrator with full access rights. It is located by default in the sys:Administrators user group.

No rights whatsoever can be set for this entity on the server.

"sys:Administrators"

This directory entity represents the user group with full access rights on the server.

As in the case of the sys:Administrator user, no rights whatsoever can be set for this entity on the server.

If a user has been assigned to this group, the rights it confers override any other rights or restrictions from any other user group or groups the user may be in.

 

“sys:CurrentUser”

This directory entity represents an abstraction of a user in the server system. In the process of creating a new entity (class, folder, subject) the sys:CurrentUser identifier is replaced with the identifier of the user creating the entity.

The user acquires the rights and prohibitions which become explicit for a user in the entity.

»sys:Server«

A directory entity represents a server user on whose behalf server services are being performed, such as long-term archiving. A server user is distinguished from other users by the fact that you can’t login to the server and perform operations on his behalf.

»sys:Everyone«

A directory entity represents a user group whose members are all directory users, and the membership of the user group can't be changed. Unlike their system groups it is possible to change access rights for this group which consequentially means that access rights are also changed for all directory users.

“Anonymous”

This directory entity represents a user connecting to the server using legacy clients that do not support user authentication. The server treats this user as anonymous and as having those rights set for an Anonymous entity.

*Example: Let’s look at an example of creating an entity on the basis of a template for which the sys:CurrentUser user has editing rights (Read and Write) in the ACL. When an entity is created using this template, all rights are transfered from its ACL and added to the entity (as explicit rights) for the user currently creating the entity. This user will therefore always have the right to edit entities which they create.*

### 3.8.3 Entity components

Directory entity components define properties and types of entities (users or user groups). These components are presented below.

 

"Id"

A 32-bit unique identifier that is automatically assigned to every entity in the process of creating an entity; it cannot be changed. This identifier is used both to obtain additional information through attributes and to calculate effective access rights in the ACL.

"Type"

An 8-bit value that represents the type of the entity in directory services;
it can have the following values:

- TYPE_GROUP: This value represents a user group.
- TYPE_USER: This value represents a user.

"Account"

This character string is a unique value that represents the user account or the name of the user group. Its value can change throughout the entire entity life cycle. The largest allowable size of the string is 256 bytes of UTF-8 characters.

"FirstName"

A character string that represents the user's first name; its value is not unique and can change throughout the entire entity life cycle.
The largest allowable size of the string is 256 bytes of UTF-8 characters.

"LastName"

A character string that represents the user's last name. Its value is not unique and can change throughout the entire entity life cycle. The largest allowable size of the string is 256 bytes of UTF-8 characters.

"Description"

This character string represents a description of an entity. Its value is not unique and can change throughout the entire entity life cycle. The largest allowable size of the string is 256 bytes of UTF-8 characters.

"Email"

This character string represents the email address of the user or user group. Its value is not unique and can change throughout the entire entity life cycle. The largest allowable size of the character string is 512 bytes of UTF-8 characters.

 

"Flags"

This 32-bit unsigned value describes the properties of the entity:

- Enabled: This property determines whether a user account is enabled or disenabled.
- AdvancedAuthenticationEnabled: This property determines whether a user account enables authentication with advanced authentication methods.
- PreSharedKeyAuthenticationEnabled: This property determines whether a user account enables authentication with advanced authentication methods that use encrpytion through a shared key.
- SRP6aAuthenticationEnabled: This property determines whether a user account enables authentication with the SRP-6 authentication method.

"AuthenticationType"

This numeric value represents the type of authentication the user can select for authentication on the server. The currently supported value is TYPE_SRP6A, which represents authentication on the server using the SRP (Secure Remote Password) protocol.

"SRP6Acredentials"

An entity representing a user can be assigned user credentials that serve as a basis for verifying the authentication of the user on the server. The user credential parameters are two character strings in UTF-8 form which represent the user password and a numerical value that represents the SRP group. The latter determines the use of prime numbers in the SRP protocol *(see chapter 3.8.5.1 SRP)*.

"SecurityClass"

This numerical value represents the security class set for the user or user group. It enables access to entities with the same or a lower security class *(see chapter 3.3.5 Access)*.

### 3.8.4 Aliases

The server enables the use of aliases for user accounts. This functionality is useful when connecting the archive system to existing authentication systems from third party providers. A directory entity can be assigned any number of aliases that users can use when logging in. Each alias must be unique (among other aliases) and cannot match any of the values of the "Account" directory entity attribute.

 

### 3.8.5 Authentication

User authentication is a part of the process performed when starting a user session with the archive server.

This process checks whether the user credentials sent by the user match the user credentials saved in the server database. If the credentials match, the user has successfully verified their identity and can perform operations on the server in line with their rights. If the credentials do not match, the server denies the session, as the user has not successfully provided verification of their identity.
The server uses the SRP protocol for authentication. This protocol is an extension of the PAKE (Password-Authenticated Key Agreement) protocol.
The PAKE protocol is an interactive method that enables two or more parties to establish a secured communications channel without using a public key infrastructure and only by using passwords *(http://en.wikipedia.org/wiki/Password-authenticated_key_agreement)*.

The PAKE protocol must fulfill the following security specifications:

- Off-line dictionary attack resistance.
- On-line dictionary attack resistance.
- Forward secrecy.
- Known-session security.

Off-line dictionary attack resistance
The attacker can be passive (only observing the protocol) or active (actively altering data transfered through the protocol). In either case, the protocol cannot be used to send data (hashes, passwords, etc.) that would enable the attacker to learn the password using rainbow tables or a brute force attack.

On-line dictionary attack resistance
If the attacker is active (actively altering data transferred through the protocol) it is practically impossible to prevent them from randomly guessing the password. Regardless of this, the PAKE protocol must ensure that the least possible damage is done in the even that an attack succeeds (in the event of a successful attack, the user can guess exactly one password). Attacks of this kind are easy to identify and prevent on the server (for example by denying a session from a specific IP address after a set number of failed authentication attempts).
Forward secrecy

 

Because there is no guarantee that a password will remain confidential over the long term, the PAKE protocol must ensure the protection of keys from past sessions, even if the password has been discovered. The protocol must also ensure that if an attacker is passively monitoring key exchange, they cannot guess the session key even if they know the password.

Known-session security

If an attack succeeds and the attacker manages to start a session with the server by impersonating a user (impersonation attack), the attacker is assumed to know the entire procedure for starting a session (including all intermediate steps performed by the server and client which are not publicly known, that is, which are not sent through the protocol).
The PAKE protocol must prevent a compromised session from threatening the security of other, already established sessions. Additional information about the PAKE protocol is available at *https://eprint.iacr.org/2010/190.pdf*.

#### 3.8.5.1 SRP

The SRP protocol is an extended version of the PAKE protocol. It prevents man-in-the-middle attackers from obtaining information.
A brute force attack could use this information to guess the password without actively altering the data exchanged between the client and the server.

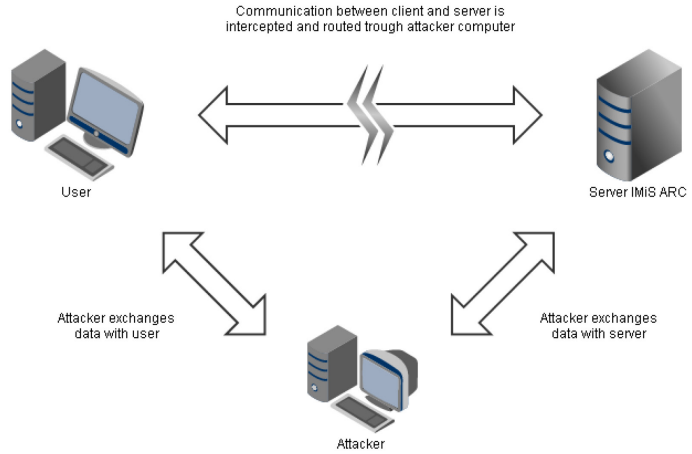


Image 29: Man-in-the-middle attack on IMiS®/ARChive Server

 

The SRP protocol has the following properties:

- Enables authentication on the server.
- Is resistant to dictionary attacks.
- Does not require public key infrastructure and consequently there is no need for trusted third parties such as issuers of digital certificates.
- The client sends the server a zero-knowledge password proof. This is an interactive method. The client proves to the server that it knows the password value without sending the server any proof (a hash of the password, etc.) of the value.

SRP (version 6) is used:

- To ensure transport layer security (a description of the SRP version can be found at *http://en.wikipedia.org/wiki/TLS-SRP*).
- In the EAP (Extensible Authentication protocol – RFC 3748).
- In the SAML protocol.

SRP is additionally a part of the following standards: IETF (RFC 2944, RFC 2945, RFC 5054), IEEE (P1363.2) and ISO (IEC 11770-4).

#### 3.8.5.1.1 Protocol

Mathematical operations in the SRP protocol are limited to those operations specified in the mathematical ring *(http://en.wikipedia.org/wiki/Ring_%28mathematics%29)*.

The SRP protocol has the following parameters:

- *N* – A secure prime number that must conform to the following formula: $N = 2 \times q + 1$, where *q* is a prime number. The prime number N determines the size of the mathematical ring and is a publicly known value.
- *g* – Multiplicative group generator (The combination of g and N is described in RFC 5054 - *http://www.tools.ietf.org/html/rfc5054*).
- *H()* – Hashing function.
- *k* – The multiplicative parameter that creates asymmetry in the SRP protocol and by doing so enhances resistance to an active attack.
- *s* – A random value known as a salt (*http://en.wikipedia.org/wiki/Salt_%28cryptography%29*) that is used to generate the password hash.
- *I* – User name.

 

- *p* – Password in plain text.
- *x* – Private key.
- *v* – Server verifier.
- *u* – Scrambling parameter.
- *a, b* – Random numbers that represent a short-term key.
- *A, B* – Public parameters.
- *S* – Session key.
- *K* – Secure session key that is the result of successful authentication
- $M_1$ – Proof that the client knows the password.
- $M_2$ – Proof that the server knows the password.

A precondition for starting the authentication process with the SRP protocol is that the server has the *v* verifier saved. It will use this verfier to check the content of the user password. The private key *x* is required to create the verifier, which is calculated the following way:

$$x = H(s, p)$$

A random value *s* is used to generate a private key and plain text password; the two are joined and a hash is calculated. Its value is the value of the private key.
The server verifier is then calculated using the following formula:

$$v = g^x$$

IMiS®/ARChive Server saves the verifier together with the random value *s*; it destroys the private key because it no longer needs it. The verifier calculated this way (together with the random value) therefore belongs to a specific user, which is why a link is required between the verifier and the user to whom the verifier belongs. On the server, this function is performed by the directory, that is, by directory services.

The authentication process may now begin. First the client and the server calculate the multiplicative parameters from the prime number and the multiplicative group generator:

$$k = H(N, g)$$

The client initiates authentication by using the secure random number generator to generate a random number *a*. This number is then used to calculate the value of the parameter *A*.
The user name *I* is then sent to the server together with the parameter *A*.
The parameter *A* is calculated using the following formula:

 

$$A = g^a$$

The server receives the user name and the parameter *A*, then uses directory services to obtain the verifier *v* and the random value *s*. The client uses the secure random number generator to calculate the value of *b*. This number is then used to calculate the value of the parameter *B*.

The parameter *B* is then sent to the server along with the random value *s*.

The parameter *B* is calculated using the following formula:

$$B = k \times v + g^b$$

The first time parameters are exchanged, both sides check the validity of *A* and *B*.

These parameters are valid if they meet the following criteria:

$A \neq 0 \ (mod\ N)$ - Checking on the server

$B \neq 0 \ (mod\ N)$ - Checking on the client

If either of the parameters does not meet the criteria, the authentication process is terminated. If the parameters meet the criteria, the authentication process continues.

The client and server calculate the scrambling parameter *u* using the following formula:

$$u = H(A, B)$$

The scrambling parameter must meet the following criteria: $u \neq 0$. If it does not, the authentication process is terminated. Authentication then continues.

The client and the server each calculate their own proof of password recognition and a secure session key *K*.

Calculating the secure session key on the client

The client takes the entered password *p* and receives the random value *s* from the server and calculates the private key *x* using the formula shown above: $x = H(s, p)$.

On the basis of the private key, a session key *S* and its hash *K*, which represents a secure session key, are calculated. The secure session key is calculated using the following steps:

$$S = (B - k \times g^x)^{(a + u \times x)}$$

$$K = H(S)$$

 

Calculating the secure session key on the server

The server calculates the secure session key using the following formulas:

$$S = (A \times v^u)^b$$

$$K = H(S)$$

The client first sends the server its proof of password recognition and proof of the secure session key. The proof is calculated using the following formula:

$$M_1 = H(H(N)\ xor\ H(g), H(I), s, A, B, K)$$

The server uses the same formula to calculate its own proof $M_1$ by including the value of the secure session key *K* in its calculation. The values of the proofs must match or an authentication error will occur and the server will deny the session.
If the values match the server calculates its own proof using the formula below and sends it to the client:

$$M_2 = H(A, M_1, K)$$

The client uses the same formula to calculate its own proof $M_2$ and includes its own secure session key K in the formula. If the values of the proofs do not match, an authentication error will occur and the server will deny the session. If the proofs match, the authentication was successful. Besides successful authentication, the client and server now also have a secure session key that they can use to encode data and set up a secure transport channel.

### 3.8.6 The directory entity life cycle

The life cycle of an entity in directory services can be divided into six types of processes:

- Opening an entity
- Creating an entity
- Editing an entity
- Saving an entity
- Discarding an entity
- Deleting an entity.

 

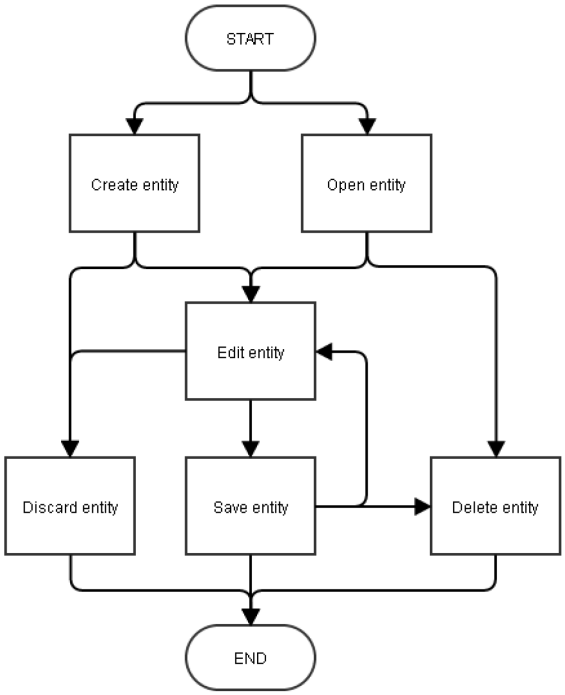


Image 30: Interconnected processes in the life cycle of an entity in the directory

#### 3.8.6.1 Opening an entity

Opening a directory entity is the process of obtaining an instance of the entity from the directory. During the initialization process, IMiS®/ARChive Server loads all entities located in the directory. At the same time, when an entity is opened it checks if the entity was previously loaded. If it was loaded, it returns its instance; if not, it returns an error.

#### 3.8.6.2 Creating an entity

The entity creation process is the first process in the directory entity life cycle.
First, the entity type is selected and a name is entered for the user account or user group.
During the editing process, additional components can be set based on the entity type.
If the newly created entity is discarded before being saved, the content of the directory is not changed.

#### 3.8.6.3 Editing an entity

Both newly created entities and entities that already exist in the directory can be edited.
When editing entities, the components described *in chapter 3.8.3 Entity components* can be set and changed.

 

#### 3.8.6.4 Saving an entity

Saving an entity is the process where all the values created or changed during the editing process are saved to the directory.

#### 3.8.6.5 Discarding an entity

An entity is discarded if the user would not like to keep a newly created entity or an entity they are editing. In both cases, the content in the directory is not changed, because when an entity is edited, that entity's last saved values remain in the directory, and a newly created entity goes into the directory only once it has been saved.

#### 3.8.6.6 Deleting an entity

Deleting an entity is done by marking the entity in the directory as deleted; it is not physically deleted from the directory. A deleted user or user group no longer exists for the server.
It still has to save their 32-bit identifiers for attribute support and to ensure the uniqueness of the 32-bit identifiers used to calculate effective rights in the ACL.

## 3.9 Backup copies and restoring data

Backup copies or backups must be regularly made to ensure effective data security.
It is important that backup copies do not fall into the hands of unauthorized persons, and that they remain unharmed in the event of an accident. Backup copying reduces the risk of data loss due to a technical malfunction of the backup media, program errors, natural disasters, unauthorized access, human error...
Backup copies enable data to be restored and returned to a previous state.

IMiS®/ARChive Server enables backup copies and restoration of the following:

- Documentary records.
- The classification scheme in its entirety or only selected classes, folders and documents of the classification scheme.
- Metadata.
- Audit trails.
- Digital certificates.
- Access Control Lists (ACLs).
- Server directory.

 

The table below shows tables in the database with a description of their content for creating backups and data restoration.

| Table | Description |
|---|---|
| ACLENTRY | Entries for the ACL. |
| ACLENTRYVALIDITY | Time limits of access rights. |
| ATTRIBUTE | Attributes defined in IMiS®/ARChive Server. |
| ATTRIBUTEVALUE | A table linking an attribute, a value and the entity to which the value belongs. |
| ATTRIBUTEVALUEBINARY | This table contains binary attribute values. |
| ATTRIBUTEVALUEDATETIME | This table contains date values with a time component. |
| ATTRIBUTEVALUEDOUBLE | This table contains a real (rational) number in a floating point with a double precision. |
| ATTRIBUTEVALUEFILE | This table contains values about files. |
| ATTRIBUTEVALUEINT32 | This table contains signed 32-bit integers. |
| ATTRIBUTEVALUEINT64 | This table contains signed 64-bit integers. |
| ATTRIBUTEVALUEINT128 | This table contains signed 128-bit integers. |
| ATTRIBUTEVALUEINT128IDX | This table contains indexed, signed 128-bit integers. |
| ATTRIBUTEVALUESTRING | This table contains unlimited strings of UTF-8 characters (they are limited by the capacity of the platform). |
| ATTRIBUTEVALUESTRING10 | This table contains strings of UTF-8 characters 10 bytes in length. |
| ATTRIBUTEVALUESTRING20 | This table contains strings of UTF-8 characters 20 bytes in length. |
| ATTRIBUTEVALUESTRING30 | This table contains strings of UTF-8 characters 30 bytes in length. |
| ATTRIBUTEVALUESTRING40 | This table contains strings of UTF-8 characters 40 bytes in length. |
| ATTRIBUTEVALUESTRING50 | This table contains strings of UTF-8 characters 50 bytes in length. |
| ATTRIBUTEVALUESTRING100 | This table contains strings of UTF-8 characters 100 bytes in length. |
| ATTRIBUTEVALUESTRING200 | This table contains strings of UTF-8 characters 200 bytes in length. |
| ATTRIBUTEVALUESTRING20IDX | This table contains indexed strings of UTF-8 characters 20 bytes in length. |
| ATTRIBUTEVALUESTRING30IDX | This table contains indexed strings of UTF-8 characters 30 bytes in length. |
| ATTRIBUTEVALUESTRING40IDX | This table contains indexed strings of UTF-8 characters |

 

| Table | Description |
|---|---|
| | 40 bytes in length. |
| ATTRIBUTEVALUESTRING50IDX | This table contains indexed strings of UTF-8 characters 50 bytes in length. |
| ATTRIBUTEVALUESTRING100IDX | This table contains strings of UTF-8 characters 100 bytes in length. |
| ATTRIBUTEVALUESTRING200IDX | This table contains strings of UTF-8 characters 200 bytes in length. |
| ATTRIBUTEVALUEUINT32 | This table contains unsigned 32-bit integers. |
| ATTRIBUTEVALUEUINT64 | This table contains unsigned 64-bit integers. |
| ATTRIBUTEVALUEUINT128 | This table contains unsigned 128-bit integers. |
| ATTRIBUTEVALUEUINT128IDX | This table contains indexed, unsigned 128-bit integers. |
| COMPRESSION | This table contains information about plug-ins for data compression. |
| CONTENT_TYPE | This table contains information about supported MIME types in IMiS®/ARChive Server. |
| DIGITALCERTIFICATE | This table contains the digital certificates located in the server's digital certificate store. |
| DIRECTORYENTRY | This table contains the entities (users or user groups) in the server directory. |
| DIRECTORYENTRYALIAS | This table contains aliases for the entities in the server directory. |
| DIRECTORYENTRYGROUP | This table contains information on user group members. |
| ENTITY | This table contains information on server entities (classes, folders, documents). |
| ENTITYCLASS | This table contains information about classes on the server. |
| ENTITYDOCUMENT | This table contains information about documents on the server. |
| ENTITYFOLDER | This table contains information about folders on the server. |
| COMPRESSIONLIB | This table contains information about libraries for data compression. |
| PROFILE | This table contains information about HSM profiles. |
| PROPERTY | This is a generic table with settings. |
| STORAGE_DRIVER | This table contains information about the drivers for HSM volumes. |
| TEMPLATE | This table contains server templates. |
| TEMPLATEATTRIBUTE | This table contains links between templates and their attributes. |

 

| Table | Description |
|---|---|
| TEMPLATEBIND | This table contains information about binding between server templates. |
| ARCHIVALINFORMATIONPACKAGE | This table contains archival inforamtion packages (AIPs) |
| ARCHIVALINFORMATIONPACKAGEQUEUE | This table represents the AIP queue. |
| CREATEAIPQUEUE | This table represents queue for creating AIPs. |
| AIPJOBTIMESTAMP | This table contains information about the time stamping process. |
| REVOCATIONDATA | This table contains information about revoked digital certificates. |
| FULLTEXTINDEXINGQUEUERECORD | This table contains information about content indexing. |
| TIMESTAMP | This table contains timestamps. |
| TIMESTAMPRULE | This table contains rules for time stamping. |
| VOLUME | This table contains information about HSM volumes. |
| LOOKUPTABLE | This table contains the statuses, significance or importance and security classes of entities. It also contains email delivery priorities and physical record statuses. |
| HASH128 | This table contains hashes 128 bits in length. |
| HASH160 | This table contains hashes 160 bits in length. |
| HASH224 | This table contains hashes 224 bits in length. |
| HASH256 | This table contains hashes 256 bits in length. |
| HASH384 | This table contains hashes 384 bits in length. |
| HASH512 | This table contains hashes 512 bits in length. |
| REVOCATIONDATATIMESTAMP | This table contains links between information about revoked digital certificates and their timestamps. |
| REVOCATIONDATADIGITALSIGNATURE | This table contains links between information about revoked digital certificates and their digital signatures. |
| TIMESTAMPARCHIVALINFORMATIONPACKAGE | This table contains Archival Inforamtion Packages (AIPs) for time stamping. |
| CERTIFICATEBODY | This table contains the binary or Base64 encoded data representing digital certificates. |
| FILEDESCRIPTION | This table contains descriptions of content. |
| ATTRIBUTEVALUENULL | This table contains entities with »null« value attributes. |
| REVIEWQUEUE | This table contains reviews that are in the preparation and implementation phase. |

Table 29: Tables with a description of data for backups and restoration

 

IMiS®/ARChive Server additionally enables the automatic generation of secure backups of all files important for operations and consistency. These files encompass the data collection where IMiS®/ARChive Server stores metadata about records, audit trail data, settings and archival record files, among other things.
As a software producer, we recommend that our customers manage secure backup copies and restoring of the archive server with a professional client/server tool (for example, IBM Tivoli Storage Manager, HP Data Protector, CA ARCserve).

### 3.9.1 Creating backup copies

A secure file storage client must be installed on IMiS®/ARChive Server to create secure backup copies. The client's location depends on which client/server tool you are using.

#### 3.9.1.1 Backup copy settings

Backup copy settings depend on which client/server tool you are using. In most cases, a description of the settings can be found in the tool's documentation.
Basically, there are three types of backup copying:

- Full backups, which encompass all data.
- Incremental backups, which encompass any data that has changed since the last backup was made.
- Differential backups, which encompass any data that has changed since the last full backup.

In combination with the dbtool tool, any of these backup types can be used with IMiS®/ARChive Server.

### 3.9.2 Restoring data

Restoring archived documents in IMiS®/ARChive Server depends on the server status and the desired results of server data restoration.
If the usefulness of the data that remains on the hard drive is in question, or if you have the option of restoring to a state prior to the malfunction of the hard drive, we advise you to contact our technical support staff at *support@imis.eu*.

Restoring data from a backup ensures the integrity of the data, including the audit trail.

 

Restoring data in IMiS®/ARChive Server depends on the server status and the desired results following data restoration. To successfully restore data, the server must at least save backup copies for:

- The server database (the default location in the directory is /iarc/db).
- Server volumes, including all content, recursively (the default location in the directory is /iarc/vol).
- The server configuration file iarc.conf (the default location in the directory is/etc).

A full backup can be achieved if, besides the minimum requirements, the backup also saves:

- The /iarc/webadmin directory and all its content, recursively.
- The /iarc/imisarc directory and all its content, recursively.
- The /etc/init.d/iarcd file.
- The /iarc/log/iarc directory and all its content.

#### 3.9.2.1 Data restoration settings

When restoring data, the first step is to obtain the most recent saved text file with data from the database and the content of the server volumes. The user should note that the access right settings for the user in whose name the server is being booted are correct (the default user is iarc, the chown or chmod tools can be used to change owner rights).

The process for restoring the volumes is as follows:

- If the old content of the volumes still exists, (the default directory location is /iarc/vol) it is deleted.
- The user copies the volume objects from the backup to the server volumes.
- Access rights (Read and Write rights) must be set to copy objects.

The server database restoration process is as follows:

- The user copies the blank server database to the server location (the default directory location is /iarc/db).
- Read and Write rights are set on the server database.
- The database text files from the backup are copied to a temporary location on the hard disk.
- Read rights are set for the text files.

 

- The server database is restored using the dbtool tool. Individual tables can be restored using the –t parameter, or the –a parameter can be used to import data to all tables for which text files exist; (*see chapter 8.3 Configuration*).

### 3.9.3 Example

When creating backups of IMiS®/ARChive Server data, the possibility of data loss or partial data corruption in the backup itself, as well as the possibility of partial data corruption, must be taken into account. Best practices therefore also include creating backup copies and storing duplicates at a remote location. Backups and any copies should be stored in a suitably secured, fireproof location. The possibility of complete data loss can also mitigated by creating a suitable plan for creating and storing backups over longer periods.

*Example: IMiS®/ARChive Server backups can be created daily.*
*The original backups are stored in a fireproof cabinet at the same location as the server and copies of the backups are stored in a safe at a remote location. The backup copies made at the end of a week, month and year are stored in the safe.*
*The duration of their storage depends on the durability of the media and the usefulness of the backed up content.*

The usefulness of backups must also be checked periodically.

Data from a backup copy is restored at another location and compared with source data from IMiS®/ARChive Server. The plan for making and storing backups must be periodically checked, or at least when essential changes occur, such as replacing servers, media or infrastructure, which directly impact events linked to IMiS®/ARChive Server. The plan is generally checked by the IT systems auditor, who issues an opinion.

### 3.9.4 Problems restoring data

If the usefulness of the data that remains on the hard drive is in question, or if you have the option of restoring to a state prior to the malfunction of the hard drive, we recommend you contact the technical support staff of the producer of IMiS®/ARChive Server.
The examples below present problems that could occur when restoring data.

 

*Problem 1:*

When restoring data by importing with dbtool, the following error is returned (the text file can have a different name and structure, depending on the tables being restored in the database):

Using 'sl_SI.UTF-8' locale settings.
Document Root: /iarc/db/
Port: 21553
attribute.txt file, open error 13: Permission denied.
Import command completed with 1 error(s).

*Solution for problem 1:* Access rights have not been correctly set for the text files.
The user in whose name the server is running does not have Read rights for the text files (the default user is iarc). The solution is correctly setting access rights for the text files and retrying the data restoration process.

*Problem 2:*

When restoring data by importing with dbtool, the following error is returned:

Using 'sl_SI.UTF-8' locale settings.
Document Root: /iarc/db/
Port: 21553
Error; Unable to move to requested record (dberr#-940) while going backwards! (file = RaimaKeyIterator.cpp, func = moveTo, line = 200)

*Solution for problem 2:* Access rights have not been correctly set for the server database.
The solution is setting Read and Write access rights in the server database for the user in whose name IMiS®/ARChive Server will be started.
Once the correct settings have been entered, restoration should proceed normally.

*Problem 3:*

When restoring data by importing with dbtool, the following error is returned (the name of text file and table values can differ, depending on the tables being restored in the database):

Using 'sl_SI.UTF-8' locale settings.
Document Root: /iarc/db/
Port: 21553
Importing attribute.txt (3194 bytes) to ATTRIBUTE ... Error; Tried to insert duplicate key into table 10008! (file = RaimaDataSet.cpp, func = InsertObject, line = 207)

*Solution for problem 3:* The server database to which the user is attempting to import the data already contains data identical to the data in the text files.

 

There are multiple solutions for this error:

- Replacing the server database with a blank one and retrying the data importation.
- The individual table for which the error described above was returned when importing data is deleted (the `init` command in the `dbtool` can be used), then the data importation process is repeated for that table (not applicable for the audit trail, which cannot be deleted).
- When importing using `dbtool`, the `–o` option can be used to overwrite any existing data in the database with data from the text files.

  *Example: A command for importing all text files with the overwrite option:*
  *su - iarc -s /bin/bash -c "cd /opt/IS/imisarc && ./dbtool -f /etc/iarc.conf -w /iarc/db/ -o -a imp"*

# 4 SYSTEM REQUIREMENTS

## 4.1 Hardware

The servers available on the market today for the most part fulfill the requirements of IMiS®/ARChive Server, as it requires few resources and therefore can also function without problems in virtual environments. It is however necessary to pay attention to the server architecture and to harmonize it with one of the architectures supported by the product. These are primarily 32- and 64-bit versions of Intel x86 platforms.

### 4.1.1 Planning server processor power

When selecting processor power, attention must be paid to the anticipated server workload (number of clients, number of parallel user sessions, average size of archived content, use of an audit trail, etc.).

In light of the current functional properties of the product, a number of mid- and high-performance processors available on the market enable a quality operating environment. The recommended requirements of the operating system usually serve as a guide when selecting processing power.
In a hypothetical system with 500 users and an average of 200 views per day, and with an average archived content size of 100 kB, even in the event of simultaneous use by all users, and even if these users all perform transactions in the same period, one Xeon QuadCore processor with a mid range frequency or one processor from the Intel Core i5/i7 family of processors should be sufficient.

 

### 4.1.2 Planning server memory capacity

When planning RAM size for IMiS®/ARChive Server, the following must be taken into account:

- Operating system requirements.
- Basic server requirements (to function, the server will need around 512 MB).
- Number of simultaneous users; every connection will require around 256 KB.
- The minimum recommended RAM size is equal to the sum of 512 MB, required by the server and the minimum RAM size required by the producer of the operating system.

The recommended RAM size is equal to the sum of the RAM needed by the operating system services plus 1024 MB (1 GB) for IMiS®/ARChive Server.

### 4.1.3 Planning server hard disk capacity

When planning server hard disk capacity, the following must be taken into account:

- Operating system requirements.
- Anticipated daily object growth increments.
- Anticipated object growth as a result of the conversion of legacy archives to digital form.
- Average object size.
- Anticipated server use time (for example, 5 years).

The objects IMiS®/ARChive Server stores in its volumes can be of different types and can come from different computing environments.
Objects scanned using IMiS®/Scan with a resolution of 300 PPI, in black and white (1 bit of color depth), using the default compression method (CCITT G4 T6) take up an average 45 KB per scanned page. The use of other compression methods typically increases color depth and resolution and, as a result, object size (see table below).

 

| Color depth | Black and white (1 bit) | Grayscale (8 bit) | Color (24 bit) |
|---|---|---|---|
| **None** | 605 KB | 5 MB | 15 MB |
| **CCITT G3** | 85 KB | x | X |
| **CCITT G4 T6** | 45 KB | x | X |
| **JBIG** | 36 KB | x | X |
| **JBIG 2bit** | x | 84 KB | X |
| **JBIG 3bit** | x | 165 KB | X |
| **JBIG 4bit** | x | 420 KB | X |
| **Packed bits** | 109 KB | 5 MB | 15 MB |
| **LZW** | 75 KB | 3.2 MB | X |
| **Packed bits 8 bit** | x | x | X |
| **Packed bits 24 bit** | x | x | X |
| **ZIP** | 56 KB | 3 MB | 9 MB |
| **Wang JPEG** | x | 315 KB | 363 KB |
| **Sequential JPEG** | x | 315 KB | 360 KB |
| **Progressive JPEG** | x | 310 KB | 334 KB |

Table 30: Average scanned document sizes using different scanning methods

When selecting a suitable compression method, keep in mind that the transfer of larger objects through the computer network requires greater bandwidth and can affect the responsiveness of the network.

We advise against the use of grayscale and color scanning, as most current scanners for capturing documents use advanced methods and filters for graphic processing to ensure the optimal quality of scanned documents.

We recommend the use of disks with appropriate data protection and scalability.
We recommend the use of up-to-date disk controllers that enable caching for reading and writing. The caching should provide autonomous charging support or should be performed using flash memory (EEPROM) techonology that can store data without electricity, unlike the static RAM older RAID drivers are equipped with.
Disks should be merged in a redundant disk array. To enhance efficiency, we recommend a RAID5 type disk array with an additional backup disk.
We advise against using disks that could be accessed by IMiS®/ARChive Server through the local network, for example NAS (Network Attached Atorage) or disks located on another server with which IMiS®/ARChive Server interacts using CIFS, NSF and similar protocols.

 

### 4.1.4 Communication channels

IMiS® clients communicate with IMiS®/ARChive Server through network port 16807 if different settings have not been entered in the /etc/iarc.conf configuration file. Communication through this port must be enabled, and rules in any firewalls or other active network programs should allow IMiS® to set up a client-server connection; server-client connections are not foreseen/necessary.

### 4.1.5 Connecting to network hardware

We recommend using redundant connections and connecting to the local network back bone with as few intermediaries as possible. It is best for the server to be connected directly to the network switch. The network protocol between IMiS® clients and IMiS®/ARChive Server is optimized for 32 KB data packets (reading/writing archived content) and smaller command packets.
An individual network packet can exceed 32 KB if the request or request reply so demand (greater data volume in a request).

All communication between the server and the client is compressed using GZIP,
ensuring high throughput. Development-phase testing and observations from large production systems revealed a bottle neck in the width of communication channels and hardware.

### 4.1.6 Administrator rights

The rights the IMiS®/ARChive Server administrator requires for their work are equivalent to those of the root user. The server administrator typically assigns rights; these rights must be sufficient for installation, upgrading and server administration.
The administrator does not require privilege rights for their work.
During installation, the IMiS®/ARChive Server installation script creates a iarc user account and an iarc user group with which all processes are launched on the server.
This means that in the event of an attack through some error in the server's application code, it will not be possible for the attacker to obtain the root user's rights.

### 4.1.7 Managing hardware operations

Most hardware manufacturers include a system health monitoring system with their servers (i. e. IBM Tivoli, HP Insight Systems Manager, Dell OpenManage).

 

The use of such a system is wise when hardware operation problems occur or when system managers require data about server operations. It is also good if the management system enables the reporting of system operation errors via mobile phones or email.

### 4.1.8 Minimum requirements

- A server with an Intel Pentium x86 ali x86_64 processor running at 800Mhz or a compatible x86 architecture (consult minimum requirements for the installation platform - operating system).
- 1GB RAM (consult the minimum requirements for the operating system and add 512 MB).
- Suitable disk capacity for the anticipated volume of archived content; a minimum 1 GB for server operations.
- Access to a network via TCP/IP protocol (IPv4 or IPv6).
- Any hardware that provides support for the operations of the Linux operating system with the above listed distributions in network mode.

### 4.1.9 Recommended requirements

- Server with an Intel Xeon E5/E7 or Xeon 5xxx/6xxx/7xxx (x86_64) multicore processor running at 2GHz (or better).
- 4GB or more high-frequency SDRAM (DDR3/DDR4).
- Fast motherboard with a high-frequency Front Side Bus (1GHz or faster).
- Volumes on RAID5 logical disks/partitions (disk space calculated 3 to 5 years in advance).
- SCSI/SAS controllers with write-back cache capabilities (up to 40% greater efficiency), a 128MB cache or larger with battery backup or flash memory (for power outages).*
- High-speed SCSI/SAS disks (10k/15k RPM) with appropriate caching.*
- Redundant power supply with a cooling system.
- Redundant network connection at 1Gbps or more with the IPv4 or IPv6 protocol.

*Note:*

** The disk subsystem can be replaced with suitable SAN network volumes that are comparable to recommended local disk capacities in terms of performance.*

** The product also functions normally in world renowned virtual environments such as VMware ESX/ESXi, Microsoft Hyper-V, Oracle VM and others if appropriate virtual resources are provided to facilitate a performance environment comparable to that achieved using the recommended hardware listed above.*

 

## 4.2 Software

### 4.2.1 Operating systems

IMiS®/ARChive Server functions on a x86/x86_64 operating system and on the following derivatives of Red Hat and SuSE distributions:

- RHEL 4.x
- RHEL 5.x
- RHEL 6.x
- CentOS 4.x
- CentOS 5.x
- CentOS 6.x
- SLES 11.x
- SLES 12.x
- OpenSuSE 11.x
- OpenSuSE 12.x.

#### 4.2.1.1 Recommended requirements

Linux OS (RedHat EL/Fedora, SuSE SLES/OpenSuSE) (all systems designed around any 2.6.x core).

When installing and running IMiS®/ARChive Server the operating system must provide the following tools and libraries: The tools and libraries of the Linux operating system can form an integral part of different operating system installation packages.

### 4.2.2 List of required system tools

bash (for more information see *http://www.linuxmanpages.com/man1/bash.1.php*)
chmod (for more information see *http://www.linuxmanpages.com/man1/chmod.1.php*)
chown (for more information see *http://www.linuxmanpages.com/man1/chown.1.php*)
cp (for more information see *http://www.linuxmanpages.com/man1/cp.1.php*)
echo (for more information see *http://www.linuxmanpages.com/man1/echo.1.php*)
grep (for more information see *http://www.linuxmanpages.com/man1/grep.1.php*)
mv (for more information see *http://www.linuxmanpages.com/man1/mv.1.php*)
ps (for more information see *http://www.linuxmanpages.com/man1/ps.1.php*)
pwd (for more information see *http://www.linuxmanpages.com/man1/pwd.1.php*)
rm (for more information see *http://www.linuxmanpages.com/man1/rm.1.php*)
rmdir (for more information see *http://www.linuxmanpages.com/man1/rmdir.1.php*)
rpm (for more information see *http://www.linuxmanpages.com/man8/rpm.8.php*)
sed (for more information see *http://www.linuxmanpages.com/man1/sed.1.php*)
sh (for more information see *http://www.linuxmanpages.com/man1/sh.1.php*)

 

su (for more information see *http://www.linuxmanpages.com/man1/su.1.php*)
touch (for more information see *http://www.linuxmanpages.com/man1/touch.1.php*)
ip (for more information see *http://www.linuxmanpages.com/man7/ip.7.php*)
ldconfig (for more information see *http://www.linuxmanpages.com/man1/ps.1.php*)
awk (for more information see *http://www.linuxmanpages.com/man1/awk.1.php*)
find(for more information see *http://www.linuxmanpages.com/man1/find.1.php*)
id (for more information see *http://www.linuxmanpages.com/man1/id.1.php*)
ipcrm (for more information see *http://www.linuxmanpages.com/man8/ipcrm.8.php*)
ipcs (for more information see *http://www.linuxmanpages.com/man8/ipcs.8.php*)
killall (for more information see http://www.linuxmanpages.com/man1/killall.1.php)
setsid (for more information see *http://www.linuxmanpages.com/man2/setsid.2.php*)
groupadd (for more information see *http://www.linuxmanpages.com/man8/groupadd.8.php*)
useradd (for more information see *http://www.linuxmanpages.com/man8/useradd.8.php*)

### 4.2.3 List of required system libraries

libc.so.6
libm.so.6
libpthread.so.0
libstdc++.so.6
libdl.so.2
libgcc_s.so.1
librt.so.1
libbz2.so.1
libz.so.1
rpmlib

### 4.2.4 Minimum requirements

Operating system: Linux OS (RedHat EL/Fedora, SuSE SLES/OpenSuSE) (all systems designed around any 2.6.x core).

# 5 INSTALLATION

The process for installation using console tools is described below.
IMiS®/ARChive Server installation can be performed by the root user (root) or any user with equivalent rights (sudo). It follows steps and is uniform for all target groups of persons installing the server.

 

## 5.1 Installation process

Installation can only be performed in an environment that meets the minimum requirements for installing one of the supported Linux distributions.

The minimum requirements can be upgraded in line with foreseen needs *(see chapter 4.1 Hardware* and *chapter 4.2 Software)*.

The IMiS®/ARChive Server installation process is simple. The steps of the process are described below.

Step 1

The user logs into the operating system console as a `root` user or enters commands as equivalent to the `root` user using the `sudo` tool. The disk drives foreseen for IMiS®/ARChive Server should be prepared and accessible in the file system in advance. `/iarc` is the default location for most server files (database, document files, etc.). Installation is performed using the `rpm` tool.

This tool is an integral part of supported Linux distributions.

Step 2

An `rpm` command is executed to install the installation package:

```
[user1@iarc ~]# sudo rpm -ivh imisarc.9.1.1406-600.0001.el4.i386.rpm
```

The package can also have a different name. This depends on the Linux distribution and version of IMiS®/ARChive Server being used.

Step 3

If installation is successful, the following message will appear (its content may differ depending on the Linux distribution being used):

```
Preparing          ########################################### [100%]
1:imisarc          ########################################### [100%]
Performing POSTINSTALLATION Actions
POSTINSTALLATION Actions Done
```

 

Step 4

The installation process creates the following directories and files:

| | |
|---|---|
| /iarc/db and | |
| /iarc/db/iarc | Directories containing server database files. |
| /iarc/fti | Directory containing the system database for searching by full document text. |
| /iarc/vol | Directory containing the volumes where IMiS®/ARChive Server saves document files. |
| /iarc/wcache | Directory the server uses for caching document files during creation or editing. |
| /iarc/rcache | Directory for caching document files for which users have submitted a view request. |
| The /iarc/work | Directory where the server saves files of a temporary nature in individual processes. |
| /opt/IS/imisarc | Contains execution files and libraries required by the server for its operations. |
| /etc/iarc.conf | Server configuration file. |
| /etc/init.d/iarcd | Server initialization script. |

## 5.2 Post-installation processes

IMiS®/ARChive Server post-installation processes can be performed by the root user or a user with equivalent rights (administrator, trained staff, etc.)

### 5.2.1 Settings for the number of simultaneously opened files

Every process in the Linux operating system requires certain rights to function properly. This is achieved by running the process with the privileges of a user account that has been assigned adequate rights. Besides installing files, the installation process also creates a user account with the name iarc.

Following installation, the maximum allowable number of simultaneously open files per IMiS®/ARChive Server user account must be set. The recommended value is 4096 simultaneously opened files.

 

This setting is entered in the /etc/security/limits.conf file by filling in the following two lines:

```
iarc    soft    nofile    4096
iarc    hard    nofile    4096
```

Rights can also be set for all users belonging to the iarc group.

This is not explicitly required:

```
@iarc    soft    nofile    4096
@iarc    hard    nofile    4096
```

The same effect can be achieved by creating a /etc/security/limits.d/iarc.conf file and entering the two lines listed above for a group or user.

The decision to use one of these approaches should be based on:

- The internal rules of system administration of the server where IMiS®/ARChive Server is being installed.
- The personal preferences of the head system administrator - custodian.

The pam_limits PAM architecture module (Pluggable Authentication Modules for Linux, *http://en.wikipedia.org/wiki/Linux_PAM*) obeys all set commands from the /etc/security/limits.d/ directory and the /etc/security/limits.conf configuration file.

### 5.2.2 Settings for automatic start up

Once installed, IMiS®/ARChive Server is set to start up automatically. Automatic server start up upon operating system start up can also be manually set:

- For RHEL and CentOS distributions, the following command is used:
  **chkconfig iarcd on**
- For SLES and OpenSuSE distributions, the following command is used:
  **chkconfig iarcd on** or **yast**
  (yast: the standard configuration tool in SLES and OpenSuSE distributions).

Automatic start up settings for server services can also be checked:

- For RHEL and CentOS distributions, the following command is used:
  **chkconfig iarcd on**
  (returns the levels on which a service is automatically started).
- For SLES and OpenSuSE distributions, the following command is used:
  **chkconfig iarcd –list** or **yast**

 

It is important that the service starts on levels 3 and 5, as the following message shows:

```
[user1@iarc ~]# sudo chkconfig iarcd –list
iarcd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

## 5.3 Testing installation and settings

Several steps can be used to check whether installation was successful:

Step 1

In the /etc/iarc.conf configuration file, in the [Log] section, set the LogLevel parameter to 7 and start the IMiS®/ARChive Server storage server service.

A message will appear:

```
[user1@iarc ~]# sudo service iarcd start
Starting IMiS/ARChive HSM Storage Server:                  [  OK  ]
```

Step 2

Use the pstree –G command to check the status of running processes and their threads.

A list is returned, part of which is shown below (the report may be different depending on the distribution):

```
[user1@iarc ~]# sudo pstree –G
init─┬─ ...
...
   ├─iarcd─┬─iarcd───7*[{iarcd}]
   │       └─iavol───{iavol}
```

This message appears if, in the iarc.conf, the [Server] section of the ConnChilds parameter in has a value of 1 and ReqThreads has a value of 7. The message displays the main process (iarcd) that manages the connection process (another iarcd ) with 7 threads and a iavol process with one thread that manages IMiS®/ARChive Server volumes.

Step 3

The netstat –tan command is used to check whether IMiS®/ARChive Server's administration and connection process at the TCP ports defined in the iarc.conf are listening for requests.

 

If the default TCP ports are set, the message will contain the following:

```
[user1@iarc ~]# netstat -tan
Proto Recv-Q Send-Q  Local Address  Foreign Address     State
...
tcp        0      0  *:16807         *:*             LISTEN
tcp        0      0  *:16808         *:*             LISTEN
```

# 6 UPGRADING

IMiS®/ARChive Server upgrading can be performed by the root user or a user with equivalent rights (administrator, trained staff, etc.)

When upgrading, some of the following processes must be performed (all these processes can be performed in the event of a larger server version upgrade, for example to a higher main build):

- Checking the consistency of the internal server database.
- Exporting internal server database data.
- Upgrading execution programs.
- Upgrading libraries (optional).
- Expanding the server database data scheme.
- Importing server database data once the scheme has been expanded.
- Managing rights and ownership in server files.

## 6.1 The upgrade process

Step 1

Before upgrading, the IMiS®/ARChive Server database must be exported and a backup must be made using the procedure described in another chapter of this documentation.

Step 2

Upgrading can be rather time consuming, depending on the number of objects stored on IMiS®/ARChive Server. Upgrading is performed with the following command:

```
[user1@iarc ~]# sudo rpm -Uvh imisarc.9.1.1406-600.0001.el4.i386.rpm
```

 

If correctly executed, the upgrading process will return a message similar to the one shown below (differences may occur as a result of different Linux distributions):

```
Starting IMiS/ARChive HSM Storage Server:                [  OK  ]
Verifying IMiS/ARChive HSM Storage Server Database (BDB edition) integrity (this may take a while depending on your object store size)...
Database is consistent!
Exporting IMiS/ARChive HSM Storage Server Database (this may take a while depending on your object store size)...
Database Export succefull! Upgrade can proceed.
Performing POSTINSTALLATION Actions
Importing exported database files (this may take a while depending on your object store size)...
Done.
```

Step 3

Following upgrading, it is a good idea to check whether the current state of IMiS®/ARChive Server's internal database has been successfully transferred by exporting the database and verifying the consistency of the entries in the database text files and checking file and volume directory ownership and the content of the `/etc/iarc.conf` file.

## 6.2 Possible complications during upgrading

Common complication 1

When attempting to upgrade, the program returns an error:

```
error: can't create transaction lock on /var/lib/rpm/.rpm.lock (Permission denied)
```

Reason for complication 1

The user performing the upgrade does not have adequate rights.

Solution for complication 1

To upgrade, a user must log in as the root user or must use a tool that provides them with rights equivalent to those of the root user (`sudo`).

Common complication 2

When attempting to upgrade, the program returns a warning:

```
Changing ownership of volume mountpoint "/iarc/vol/vol00" recursively to iarc:iarc (this may take a while).
WARNING: Operation failed. You will need to grant access to directories and objects for user iarc group iarc manually.
```

 

Reason for complication 2

When managing rights, the volume located at /iarc/vol/vol00 could not be found or some other reason prevented the root user from changing file and directory volume ownership.

Solution for complication 2

The disk drives containing the missing volumes (in this case, they include /iarc/vol/vol00) must be connected and ownership of the directories and files must be manually set using the following command:

[user1@iarc ~]# sudo chown <iarc uporabnik>:<iarc skupina><pot> -R

In this example:

[user1@iarc ~]# sudo chown iarc:iarc /iarc/vol/vol00 -R

Common complication 3

When attempting to upgrade, the program returns an error:

ERROR: IMiS/ARChive Storage Server BDB Database consistency check reported an error in one of the database entities. Manually run '/opt/IS/imisarc/dbtool -a check' from directory /iarc/db to get extended error information. IMiS/ARChive upgrade can proceed only when database is consistent. You need to manually verify and remove any inconsistency of the database. UPGRADE ABORTED!

Reason for complication 3

IMiS®/ARChive Server's internal database cannot be accessed or is corrupted.

Solution for complication 3

Check whether the directory where the server database is located exists. Also check that the directory is not empty and that rights for it have been set for the user performing the server processes.

Then run the following command:

[user1@iarc ~]# sudo su – iarc –s /bin/bash –c "cd /opt/IS/imisarc && ./dbtool –f <path to server configuration file> –h <path to internal database> -w <path to internal database> -a check

In this example:

[user1@iarc ~]# sudo su – iarc –s /bin/bash –c "cd /opt/IS/imisarc && ./dbtool –f /etc/iarc.conf –h /iarc/db -w /iarc/db -a check

The command will return a variety of data on the database malfunction. If the error falls outside the knowledge of the administrator of the product, other channels may be used as specified in the maintenance agreement or other agreements. In this case, the manufacturer will advise the user on resolving the error or provide the use of its maintenance staff.

 

# 7 REMOVAL

The process for removing (uninstalling) IMiS®/ARChive Server can be performed by the root user or by a user with equivalent rights (administrator, trained staff, etc.)

## 7.1 The Removal process

Step 1

Check the installed version of IMiS®/ARChive Server in the rpm database (message may differ depending on the distribution used):

```
[user1@iarc ~]# sudo rpm –q imisarc
imisarc-9.1.1406-600.i386
[user1@iarc ~]#
```

Step 2

Stop the IMiS®/ARChive Server server service.

This action is also implicitly performed upon removal if the program detects that the service is running.

Step 3

Use the rpm command to uninstall the IMiS®/ARChive Server installation package. The full name from the rpm database must be entered, as shown in step 1:

```
[user1@iarc ~]# sudo rpm –e imisarc-9.1.1406-600.i386

This uninstall action WILL NOT:
  * remove IMiS/ARChive configuration file (e.g.: /etc/iarc.conf)
  * remove IMiS/ARChive database files
  * remove IMiS/ARChive log files (location set in /etc/iarc.conf)
  * remove IMiS/ARChive pid file (location in /var/run/iarc or overriden in /etc/iarc.conf)
  * remove IMiS/ARChive stored objects (on all your volume mountpoints)
  * remove IMiS/ARChive process user (iarc) and group (iarc) accounts from /etc/passwd and /etc/group

Above actions should be performed manually if required!

Uninstall complete.

[user1@iarc ~]#
```

 

# 8 ADMINISTERING THE PRODUCT

IMiS®/ARChive Server administration can be performed by the root user or a user with equivalent rights (administrator, trained staff, etc.).

Due to the ease of use, following settings can be managed by the IMiS/Client:

- User and user group access rights to entities and user defined attributes.
- User defined attributes.
- Attributes value range.
- Entity tree structure in the classification scheme and the method of setting the classification code for entities at certian tree level.
- Users and user groups with information about user, authentication, roles and group members.
- Templates with user defined attributes.
- Profiles.
- Volumes.

Detailed information are available *in chapter Server Configuration in IMiS®/Client Manual*.

## 8.1 Starting and shutting down

An initialization script is used to start and shut down IMiS®/ARChive Server. The `iarcd` script is located in the `/etc/rc.d/init.d` directory for the RHEL and CentOS distributions and in the `/etc/init.d` directory for the SLES and OpenSuSE distributions.
The initialization script is used together with the `service` tool in the following way:

```
[user1@iarc ~]# sudo service iarcd <command>
```

The valid values for the <command> option in the initialization script are:

start This command starts up IMiS®/ARChive Server.

If start up is successful, the script returns the following message:

```
Starting IMiS/ARChive HSM Storage Server: [  OK  ]
```

If start up is not successful, the script returns the following message:

```
Starting IMiS/ARChive HSM Storage Server:  [FAILED]
```

 

stop This command shuts down IMiS®/ARChive Server operations.

If shut down is successful, the script returns the following message:

```
Shutting down IMiS/ARChive HSM Storage Server:      [  OK  ]
```

If shut down is not successful, the script returns the following message:

```
Shutting down IMiS/ARChive HSM Storage Server:      [FAILED]
```

restart This command restarts IMiS®/ARChive Server. This is actually a command sequence with the `start` and `stop` commands, which is why the messages on the console are identical to those that would appear if both commands were executed.

status This command shows the status of IMiS®/ARChive Server. If the program is running, it also returns two process identification numbers:

```
Status of IMiS/ARChive HSM Storage Server: iarcd (pid 6222 6216) is running ...
```

If the service has stopped running, it returns the following:

```
Status of IMiS/ARChive HSM Storage Server: iarcd is stopped
```

For more information about potential problems starting IMiS®/ARChive Server *see chapter 9 Troubleshooting*.

## 8.2 Logging operational events

Event logging is intended primarily to check the functioning of the server. It is conducted occasionally or as needed by the server administrator or the IMiS®/ARChive Server administrator. IMiS®/ARChive Server logs events based on the logging level settings in the `/etc/iarc.conf` configuration file. The default location of the logs is `/var/log/iarc`. The active log where IMiS®/ARChive Server logs current events is located at `/var/log/iarc/iarc.log`. Older events are saved in archive log files, which are created as needed in accordance with the settings. Archive log files are created using the `/var/log/iarc/iarc.XX.log` addressing scheme (XX = sequence of the archive file, a larger number indicates events that date further back in time). Logging in IMiS®/ARChive Server is performed using the FILO principle (FILO – first in/last out).

The number and size of the archive logs can be set in the `/etc/iarc.conf` configuration file, in the `[Log]` section. The time intervals at which information is removed from the log are adjusted using the setting for the number and size of logs based on the amount of entries. Best practices dictate saving level 6 information for at least three months.

 

There are 7 logging levels. Every level represents a greater degree of detail of information being logged by IMiS®/ARChive Server.

Level 0 – Emergency

Entries of the "zero" level are errors that prevent the continued operations of IMiS®/ARChive Server. They represent serious errors, possibly in connection with data corruption.
The server therefore shuts down once it detects an error of this kind; continued use is not possible without intervention from the server administrator.
The causes of these errors are usually of an external nature, such as the failure of key server hardware components.

Level 1 – Alert

Records of the first level contain events that do not necessarily result in IMiS®/ARChive Server ceasing operations. It will cease operations if continued use could lead to a malfunction in the internal server database or the objects.

There are several possible reasons for errors on this level: hardware failure, incorrect performance of operating system functions has been detected, the server is overloaded or another program is interfering with the IMiS®/ARChive Server environment, an attempt to configure profiles and volumes has failed, etc.

Level 2 – Critical

Records of level 2 events contain errors that cause IMiS®/ARChive Server to cease operations, as continued use could lead to a malfunction in the internal server database or the objects.
There are several possible reasons for errors on this level: incorrect performance of operating system functions has been detected, insufficient server resources, the server is overloaded or an error has been detected in server operations, etc.

Level 3 – Error

Records of level 3 events contain errors where IMiS®/ARChive Server detected an operational error that is not critical and does not imply the possibility of data corruption.
The causes of these errors can pertain to the client or result from incorrect or unsuitable server setting parameters or incorrect entries in the database.

 

Level 4 – Warning

Records of level 4 events are warnings where IMiS®/ARChive Server detected an impropriety that does not essentially impact server operations and that in most cases results from unregulated client requests, and less frequently from incorrect entries in the server database or configuration file.

Level 5 – Notice

Level 5 records are records about important regular (normal) events in IMiS®/ARChive Server that could potentially be of interest to administrators.

Level 6 – Info

Level 6 records are records about less important regular (normal) events in IMiS®/ARChive Server that could potentially be of interest to administrators.

Level 7 – Debug

Level 7 entries are extended entries about all events in IMiS®/ARChive Server. These entries are used when collecting highly detailed information about server operations in the event that the causes of an error or warning are not immediately evident.

The IMiS®/ARChive Server must pay attention to messages from levels 4 to 0, as they could reveal problems in server operations and communications with clients.

## 8.3 Configuration

Configuration is performed by the `root` user or by a user with rights equivalent to those of the root user using the `sudo` tool to perform actions with IMiS®/ARChive Server user credentials. Otherwise an internal database malfunction could occur.

 

### 8.3.1 Foreseen tasks

Console tools for work with the IMiS®/ARChive Server internal database include the following:

dbtool

This tool enables the server administrator to manage the internal database.

The database exporting process can only be executed on a running server.

The database importing process can only be executed when the server is not running.

Syntax:

usage: dbtool [-f db_config_file] [-q(uiet)] [-w working_dir] [-v version] [-o override][-a | -t tables[:name],...] exp | imp | init | del

Tables:
acle - acl entry table
aclv - acl entry validity table
at - attribute table
ag - attribute group table
av - attribute value table
ab - attribute binary table
dt - attribute date-time table
db - attribute double table
fi - attribute file table
i4 - attribute int32 table
i8 - attribute int64 table
i16 - attribute int128 table
i16i - attribute int128 index table
sm - attribute max string table
s10 - attribute value string 10
s20 - attribute value string 20
s30 - attribute value string 30
s40 - attribute value string 40
s50 - attribute value string 50
s100 - attribute value string 100
s200 - attribute value string 200
s20i - attribute value string 20 index
s30i - attribute value string 30 index
s40i - attribute value string 40 index
s50i - attribute value string 50 index
s100i - attribute value string 100 index
s200i - attribute value string 200 index
u4 - attribute value uint32
u8 - attribute value uint64
u16 - attribute value uint128
u16i - attribute value uint128 index
au - audit
cx - compression
mi - content type
cnt - counter table

 

dsc - digital certificate
cbdy - certificate body
ajts - aip job timestamp
dsi - digital signature
de - directory entry
dea - directory entry alias
dg - directory entry group
ent - entity
cl - entity class
do - entity document
fl - entity folder
lo - lookup table
h128 - 128 bit hash table
h160 - 160 bit hash table
h224 - 224 bit hash table
h256 - 256 bit hash table
h384 - 384 bit hash table
h512 - 512 bit hash table
rets - revocation data/timestamp bind table
resig - revocation data/digital signature bind table
tsaip - timestamp/aip bind table
cxl - compression library
pr - profile
prop - property
sd - storage driver
tm - template
ta - template attribute
tb - template bind
aip - archival information package
aipq - archival information package queue
caq - create aip queue
revo - revocation data
fti - full text indexing queue record
ts - timestamp
tsr - timestamp rule
vo – volume
fd - file description
null - attribute value null value
revq - review queue

GetStorageInfo

This is a tool for configuring profiles and volumes and data about storage capacity.

Syntax:

usage: GetStorageInfo [ path-to-iarc.conf]

(together with the file path if the file is not located in the default location – /etc ].)

 

### 8.3.2 Configuration processes with console tools

The internal server database can be exported using the following command (the specifics depends on the configuration of the server):

[user1@iarc ~]# **sudo su - iarc -s /bin/bash -c "cd /opt/IS/imisarc && ./dbtool -f /etc/iarc.conf -w /iarc/db/ -a exp"**

```
Using 'sl_SI.UTF-8' locale settings.
Document Root: /iarc/db/
Port: 21553
Exporting ACLENTRY (3 records) to aclentry.txt ... OK.
Exporting ACLENTRYVALIDITY (0 records) to aclentryvalidity.txt ... OK.
Exporting ATTRIBUTE (52 records) to attribute.txt ... OK.
Exporting ATTRIBUTEGROUP (0 records) to attributegroup.txt ... OK.
Exporting ATTRIBUTEVALUE (0 records) to attributevalue.txt ... OK.
Exporting ATTRIBUTEVALUEBINARY (0 records) to attributevaluebinary.txt ...
OK.
Exporting ATTRIBUTEVALUEDATETIME (0 records) to attributevaluedt.txt ... OK.
Exporting ATTRIBUTEVALUEDOUBLE (0 records) to attributevaluedouble.txt ... OK.
Exporting ATTRIBUTEVALUEFILE (0 records) to attributevaluefile.txt ... OK.
Exporting ATTRIBUTEVALUEINT32 (0 records) to attributevalueint32.txt ... OK.
Exporting ATTRIBUTEVALUEINT64 (0 records) to attributevalueint64.txt ... OK.
Exporting ATTRIBUTEVALUEINT128 (0 records) to attributevalueint128.txt ...
OK.
Exporting ATTRIBUTEVALUEINT128IDX (0 records) to attributevalueint128idx.txt ... OK.
Exporting ATTRIBUTEVALUESTRING (0 records) to attributevaluemaxstring.txt ... OK.
Exporting ATTRIBUTEVALUESTRING10 (0 records) to attvalstr10.txt ... OK.
Exporting ATTRIBUTEVALUESTRING20 (0 records) to attvalstr20.txt ... OK.
Exporting ATTRIBUTEVALUESTRING30 (0 records) to attvalstr30.txt ... OK.
Exporting ATTRIBUTEVALUESTRING40 (0 records) to attvalstr40.txt ... OK.
Exporting ATTRIBUTEVALUESTRING50 (0 records) to attvalstr50.txt ... OK.
Exporting ATTRIBUTEVALUESTRING100 (0 records) to attvalstr100.txt ... OK.
Exporting ATTRIBUTEVALUESTRING200 (0 records) to attvalstr200.txt ... OK.
Exporting ATTRIBUTEVALUESTRING20IDX (0 records) to attvalstr20idx.txt ... OK.
Exporting ATTRIBUTEVALUESTRING30IDX (0 records) to attvalstr30idx.txt ... OK.
Exporting ATTRIBUTEVALUESTRING40IDX (0 records) to attvalstr40idx.txt ... OK.
Exporting ATTRIBUTEVALUESTRING50IDX (0 records) to attvalstr50idx.txt ... OK.
Exporting ATTRIBUTEVALUESTRING100IDX (0 records) to attvalstr100idx.txt ... OK.
Exporting ATTRIBUTEVALUESTRING200IDX (0 records) to attvalstr200idx.txt ... OK.
Exporting ATTRIBUTEVALUEUINT32 (0 records) to attvaluint32.txt ... OK.
Exporting ATTRIBUTEVALUEUINT64 (0 records) to attvaluint64.txt ... OK.
Exporting ATTRIBUTEVALUEUINT128 (0 records) to attvaluint128.txt ... OK.
Exporting ATTRIBUTEVALUEUINT128IDX (0 records) to attvaluint128idx.txt ... OK.
Exporting audit log data (11 records) to auditlog.bin ... OK.
Exporting COMPRESSION (1 records) to compression.txt ... OK.
Exporting CONTENT_TYPE (1334 records) to objtype.txt ... OK.
Exporting DIGITALCERTIFICATE (2 records) to digitalcert.txt ... OK.
Exporting COUNTER (0 records) to counter.txt ... OK.
Exporting DIGITALSIGNATURE (0 records) to digitalsig.txt ... OK.
```

 

```
Exporting DIRECTORYENTRY (10 records) to direntry.txt ... OK.
Exporting DIRECTORYENTRYALIAS (3 records) to direntryalias.txt ... OK.
Exporting DIRECTORYENTRYGROUP (6 records) to direntrygroup.txt ... OK.
Exporting ENTITY (26 records) to entity.txt ... OK.
Exporting ENTITYCLASS (6 records) to entityclass.txt ... OK.
Exporting ENTITYDOCUMENT (0 records) to entitydocument.txt ... OK.
Exporting ENTITYFOLDER (0 records) to entityfolder.txt ... OK.
Exporting COMPRESSIONLIB (1 records) to compresslib.txt ... OK.
Exporting PROFILE (2 records) to profile.txt ... OK.
Exporting PROPERTY (2 records) to property.txt ... OK.
Exporting STORAGE_DRIVER (1 records) to store.txt ... OK.
Exporting TEMPLATE (20 records) to template.txt ... OK.
Exporting TEMPLATEATTRIBUTE (74 records) to templateatt.txt ... OK.
Exporting TEMPLATEBIND (17 records) to templatebind.txt ... OK.
Exporting ARCHIVALINFORMATIONPACKAGE (0 records) to aip.txt ... OK.
Exporting ARCHIVALINFORMATIONPACKAGEQUEUE (0 records) to aipq.txt ... OK.

Exporting CREATEAIPQUEUE (0 records) to createaipqueue.txt ... OK.
Exporting AIPJOBTIMESTAMP (0 records) to aipjobts.txt ... OK.
Exporting REVOCATIONDATA (0 records) to revodata.txt ... OK.
Exporting FULLTEXTINDEXINGQUEUERECORD (0 records) to ftiqueuerecord.txt ... OK.
Exporting TIMESTAMP (0 records) to timestamp.txt ... OK.
Exporting TIMESTAMPRULE (0 records) to timestamprule.txt ... OK.
Exporting VOLUME (2 records) to volume.txt ... OK.
Exporting LOOKUPTABLE (3 records) to lookup.txt ... OK.
Exporting HASH128 (0 records) to hash128.txt ... OK.
Exporting HASH160 (0 records) to hash160.txt ... OK.
Exporting HASH224 (0 records) to hash224.txt ... OK.
Exporting HASH256 (0 records) to hash256.txt ... OK.
Exporting HASH384 (0 records) to hash384.txt ... OK.
Exporting HASH512 (0 records) to hash512.txt ... OK.
Exporting REVOCATIONDATATIMESTAMP (0 records) to revodatats.txt ... OK.
Exporting REVOCATIONDATADIGITALSIGNATURE (0 records) to revodatadsig.txt ... OK.
Exporting TIMESTAMPARCHIVALINFORMATIONPACKAGE (0 records) to tsaip.txt ... OK.
Exporting CERTIFICATEBODY (2 records) to certbody.txt ... OK.
Exporting FILEDESCRIPTION (2 records) to filedesc.txt ... OK.
Exporting ATTRIBUTEVALUENULL (0 records) to attvalnull.txt ... OK.
Exporting REVIEWQUEUE (0 records) to reviewqueue.txt ... OK.
Export command completed with no errors.
```

*Note: The numbers (XX records) listed in each line can change depending on the number of records.*

The result is 70 text files that represent individual tables in the database, a binary file that represents the audit trail and additional data saved in files without extensions. The files start with a letter followed by a sequence number – for example: »p0« for blob from the PROPERTY table, that contain a part of the internal IMiS®/ARChive Server base.

 

## 8.4 Administration

Administrative tasks can fundamentally affect product performance.
Correct system installation and configuration can ensure stable and expected product performance with little or no maintenance; an incorrect or faulty configuration, on the other hand, can compromise the system and make it unstable, slow or vulnerable to security threats. That is why interventions in administration of the product must be restricted to trained administrators with a detailed knowledge of the manufacturer's instructions and general best practices in planning and maintaining IT systems. We recommend entering an agreement with the manufacturer upon purchase of the product to ensure flawless, uninterrupted operation of the archive system, which, for many users, constitutes a "mission critical" system.
The administrator sets parameters for IMiS®/ARChive Server in the configuration file based on server resource usage, which they then monitor through periodic overviews of server status, log entries and requirements of the application environment.

The administrator manages the system's configuration parameters in line with the manufacturer's instructions. These parameters affect the classification scheme, entity templates, attribute metadata schemes, plug-in settings for different support services for the system, etc. In the current build, these parameters can be changed using console tools and a knowledge of the system's data model; future builds will enable this functionality via an administrator interface. We also offer product training for user-administrators and recommend customers consider a maintenance agreement, as we provide professional product management services. These services can ensure the uninterrupted operation of the server over an extended period of time.

Before any changes are made, it is a good idea to make a backup of the internal server database and configuration files, as it could be necessary to restore the old settings due to changes incorrectly made to the settings and/or the database.

### 8.4.1 The iarc.conf configuration file

Configuration parameters for IMiS®/ARChive Server can be set in the `/etc/iarc.conf` file.
In the descriptions below, the default values are used. Parameters are listed by sections according to their purpose, and are then discussed in detail.

 

They are divided according to the following principle:

Key: Description (an optional range of valid, minimum, maximum and recommended values).

Section [Server]

Path: Designates the absolute path to the server execution files (programs) and libraries. The default value is /opt/IS/imisarc.

ConnChilds: Designates the number of simultaneous connection processes on the server. The default and recommended value is 1. This value can be increased if the number of requested simultaneous client sessions exceeds 1024 for every 1024 new sessions. This value has no upper ceiling, but values over 10 are not advised from a performance standpoint and can cause problems.

ReqThreads: Designates the number of threads executing requests. The default value is 7 and there is no maximum value. The recommended value is the square of the number of threads that the server's processor is capable of executing at once and depends on the number of processor cores.

StatisticsCycle: Designates the number of seconds between the calculation of statistics that IMiS®/ARChive Server keeps on its operations. The default and recommended value is 180000. Setting a lower value can affect server responsiveness. The lowest value is 1, and the highest is 16777216.

ClientTimeout: Designates the amount of time, in seconds, a logged in client must be inactive before the server terminates the session. The default value is 3600 seconds. The smallest value is 1800 and the largest is 86400.

NoAuthClientTimeout: Designates the amount of time, in seconds, a non-logged-in (anonymous) client must be inactive before the server terminates the session. The default value is 120 seconds. The smallest possible value is 5 and the largest is 3600. Only the most basic, least sensitive information can be exchanged with the server during an anonymous session (archive name, description, etc.)

IdentPassword: Designates an encrypted compressed value of the password used by the server in encryption processes. If internal identifiers for entities in other systems are permanently stored, this value cannot be changed after the first entity has been archived, as this value affects the encryption algorithm for generating internal entity identifiers.

 

Port: Designates the number of TCP ports at which the server's connection process is listening for requests from clients. The default value is 16807.

Listen: Designates the network address to which the server connects the TCP ports where it is listening for requests from clients.
Values must conform to the IPv4 or IPv6 addressing scheme.

Examples of valid values:
192.168.92.32
fd00:192:168:92::32
192.168.92.32:16807
[fd00:192:168:92::32]:16807
[fd00:192:168:92:2340:efa1:1244:32]
fd00:192:168:92:2340:efa1:1244:32
[fd00:192:168:92:2340:efa1:1244:32]:12345
[::ffff:192.168.92.12]
::ffff:192.168.92.12
[::ffff:192.168.92.12]:65743
localhost
[::]

CXLib: Designates the relative path to libraries containing compression methods for different client platforms within the path designated by the value of the Path parameter described above.

PartialTimeout: Designates the amount of time, in seconds, in which the client must respond to a request from the server. Once this time has passed, the server closes the session. The smallest value is 1, and the largest is 60; the recommended value is 5.

PidPath: Designates the path to the ID file of the main server process.
The default value is /var/run/iarc.

CountryLanguage: Designates regional settings in POSIX format: xx_YY[.CHARSET[@variant]]; the first two letters (xx) represent the language code in the ISO-639 standard, and the second letters (YY) represent the country code in the ISO-3166 standard. CHARSET (optional) defines the code table or the code order (a list can be found in the /usr/share/i18n/charsets directory), and variant (optional) represents the national features of the language. A list of possible settings can be obtained by using the locale –a command. This value defines a range of rules used by the server to execute all operations with character strings, which can be "sensitive" to different national characters (sorting, translating, compiling, etc.) The default value is "", which effectively specifies the rules set for the system (see the locale system command *http://www.linuxmanpages.com/man1/locale.1.php*).
Special feature: To ensure the transferability of the data saved in the

 

internal database, the service internally uses UTF-8 as its internal code table regardless of the regional settings. It saves other regional settings (language, country, special language features).

DataPath: Designates the space on the disk where the server saves permanent data linked to the archive such as statistical data about transactions, etc. The default value is /iarc/db.

MaxRequestSize: Designates the largest possible one-off request from a client. The default value is 3276800 bytes. The smallest possible value is 32768 and the largest is 32768000.

[Database] section

ProviderBootstrapFile: Designates the relative path to the configuration file of the plug-in for the database service. It is relative and depends on the value of the [Server].Path parameter.

[Cache] section

ReadPath: Designates the path to the directory that represents the cache where IMiS®/ARChive Server temporarly stores accessed content to achieve faster delivery of objects to clients. Rights for the path must be arranged in such a way that a user executing server processes can read and write files there.

ReadSize: Designates the smallest possible cache value. The server dynamically adjusts this limit as needed; it does not make sense to change this value.

EditPath: Designates the path to the directory that represents the cache where the server temporarily dumps objects delivered by clients. Rights for the path must be arranged in such a way that a user executing server processes can read and write files there.

EditSize: Designates the smallest cache value. The server dynamically adjusts this limit as needed; it does not make sense to change this value.

[Log] section

LogFile: Designates the basic name (and path) of the log file where IMiS®/ARChive Server records events. Rights for the path must be arranged in such a way that a user executing server processes can write files and create new files if necessary.

MaxSize: Designates the maximum size of one log file, in bytes. The default and recommended value is 1000000, the smallest value is 65536 and the largest is 2147483648.

 

BackupCount: Designates the number of archive log files in which the server logs events using the FILO (first in-last out) algorithm. The default value is 1, the recommended value is 9.

LogLevel: The level of events the server logs in the log files (*see chapter 8.2 Logging operational events*). The smallest value is 1 and the largest is 7; the recommended value is 6.

[AuditLog] section

Enabled: The value 1 enables the logging of operations with objects, and the value 0 prevents the logging of operations with objects.
IMiS®/ARChive Server logs events in encrypted form in the internal database.

Events: Designates the range of operations the server logs as events in the log.
Valid range of operations:
ent.create (creating an entity)
ent.update (saving an entity)
ent.delete (deleting an entity)
ent.move (moving an entity in the classification scheme)
ent.openro (opening an entity for reading)
ent.openrw (opening an entity for reading and writing)
ent.acl (changing access rights in an entity with a list of the changes)
ent.prop (changing metadata values of an entity with a list of the names of attributes being changed)
ent.prm (changing metadata values for physical record storage with a list of names and values of all physical record storage attributes)
ent.secclass (changing the security class in an entity with the previous and new security class values and a reason for the change)
auditlog.query (query to view the audit trail)
dir.groupmem (changing membership of the listed user group with a list of changes (added, removed))
dir.entauth (changing log in data of the listed directory entity - user)
dir.entident (changing the reference information of the listed directory entity - user (name, surname, description, etc.))
dir.entstate (changing the state of the listed directory entity - user (active, inactive, deleted, etc.))
file.openro (opening content for reading)
file.openrw (opening content for reading and writing)
file.create (creating content)
file.delete (deleting content)

 

file.update (saving changes to content)
file.mdchange (changing metadata of content)

RequiredParams: The set of required data that must be sent by the client when opening a session or event. If one of the requested data is missing, IMiS®/ARChive Server will not open the session or event.
The valid selection of required data can contain at least one of the following:
username (name of the user performing the operation)
computername (name of the computer where the operation originated)
message (a reason/message entered by the user when performing the operation)
The default setting is username, computername.

AuthCryptoModes: Designates the selection of available cryptographic methods which the server allows the user to use to encrypt authentication messages and later communications with a client attempting to view the audit trail. The identifier represents a combination of the algorithm, key length and cryptographic method for processing packet data.

Valid values:
aes-256-cbc
aes-256-ecb
aes-256-ofb
aes-256-cfb
aes-192-cbc
aes-192-ecb
aes-192-ofb
aes-192-cfb
aes-128-cbc
aes-128-ecb
aes-128-ofb
aes-128-cfb

AuthPreSharedKey: The cached server key used for encrypting authentication meassages and later for communications with the client that represents the person authorized to view the audit trail of logged events linked to sessions and/or objects.

[Authentication] section

Methods: Designates the possible methods for establishing a session between an IMiS® client and IMiS®/ARChive Server. Valid values:
basic
advanced
psk
srp6a

 

Basic represents the older method for starting a session with the server, without delivering information about the client.
With the advanced method, a more complex HMAC method is used to start the session with the server, and foresees the use of required and non-required metadata about the client.
With the public shared key or psk advanced method, an encrypted exchange of network authentication packets ensures additional security.
The use of user credetnials or the srp6a method enables individual user to log in by obtaining user data on the basis of successful user authentication via internal sources (the directory).

CryptoModes: Designates possible encryption methods used by the server for encrypted communications with the client. The identifier represents a combination of the algorithm, key length and cryptographic method for processing packet data. Valid values:

aes-256-cbc
aes-256-ecb
aes-256-ofb
aes-256-cfb
aes-192-cbc
aes-192-ecb
aes-192-ofb
aes-192-cfb
aes-128-cbc
aes-128-ecb
aes-128-ofb
aes-128-cfb

PreSharedKey: The cached server key used for encrypting authentication messages (with the psk log in method) and later for communications with the client that represents a normal,
non-priviliged IMiS® client.

RequiredParams: The set of required data that must be sent by the client when opening a session or event. If just one of the requested data is missing, IMiS®/ARChive Server will not open the session or event.
The valid selection of required data can contain at least one of the following:
username (name of the user performing the operation)
computername (name of the computer where the operation originated)
message (a reason/message entered by the user when performing the operation)
This setting is blank by default – none of the possible options.

 

# 9 TROUBLESHOOTING

In the event of problems, it is important that administrators and users act appropriately.
Any non-expert interference with IMiS®/ARChive Server could worsen the situation and make resolving errors more difficult. Users/Administrators must be familiarized with the correct way to use the product and must act in accordance with the user documentation.
We recommend that in the event that problems occur, they turn to competent experts within the organization (system administrators). We advise system administrators to use the documentation to determine where an error has occurred and, if needed, to consult our specialists for further steps.

## 9.1 How can problems be avoided?

Regular, periodic check ups of the operations of IMiS®/ARChive Server are crucially important for the timely detection of any problems or errors in the operations of the program.
These check ups include check ups of the consistency of the disk system (the independent disk or disk array) and file system. Problems with the disk system can be avoided by selecting reliable hardware and ensuring that the disks used are locally connected to the server with adequate redundancy.
NAS disk systems should be avoided, as should the shared use of disks on other servers or disks accessible through the local server.

The consistency of the internal server database should also be checked periodically.
For more information *see chapter 8.3 Configuration*.

The optional maintenance agreement is also of key importance. It ensures minimum response times in the event of more serious errors or system outages.

## 9.2 Frequent problems

Frequent problem 1

When attempting to view an object saved on IMiS®/ARChive Server, IMiS®/Scan or IMiS®/View clients return “Error 61523”.
Other clients (IMiS®/Storage Connector, for example) return the following error:

> IMiS/ARC Client <IASession.Open> Failed to establish connection to the cluster node <10.1.1.10, 16807> (Reason: Error <TimedOut> occurred while opening network connection.).

 

The server is accessible on the network, but the service at the port where it is listening cannot be accessed (checking using the `telnet` program).

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=3 ttl=64 time=0.186 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=4 ttl=64 time=0.183 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=5 ttl=64 time=0.164 ms


--- iarc.acme.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.164/0.450/1.530/0.540 ms

[user1@test ~]# telnet iarc.acme.com 16807
Trying 10.0.0.10...
.. (long pause) ...
telnet: connect to address 10.1.1.10: Connection timed out
[user1@test ~]#
```

The server is running. This can be checked using a console command on the server where it is installed:

```
[user1@iarc ~]# sudo service iarcd status
Status of IMiS/ARChive HSM Storage Server: iarcd (pid 23209 23203) is running...
[user1@iarc ~]#
```

Cause of problem 1

The firewall between the client and the server on the server or network is preventing IMiS® clients from communicating with IMiS®/ARChive Server through TCP port 16807 or a different port (depending on the settings in the `/etc/iarc.conf` file).

Solution for problem 1

The firewall must be reconfigured to allow communication between IMiS® clients and the server.

Frequent problem 2

When attempting to save a new object, the server returns the following message:

> You cannot create an entity with template »%TEMPLATE_NAME%« under specified parent since it's not included in the list of allowed templates.

 

The server is accessible on the network, and the service at the port where it is listening is responding (checking using the telnet program).

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=3 ttl=64 time=0.186 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=4 ttl=64 time=0.183 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=5 ttl=64 time=0.164 ms


--- iarc.acme.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.164/0.450/1.530/0.540 ms

[user1@test ~]# telnet iarc.acme.com 16807
Trying 10.1.1.10...
Connected to iarc.acme.com.
Escape character is '^]'.
Connection closed by foreign host.
[user1@test ~]#
```

The server is running. This can be checked using a console command on the server where it is installed:

```
[user1@iarc ~]# sudo service iarcd status
Status of IMiS/ARChive HSM Storage Server: iarcd (pid 23209 23203) is running...
[user1@iarc ~]#
```

Cause of problem 2

The client is attempting to save an object with an existing template that has not been classified in the list of allowed templates for the parent entity.

Solution for problem 2

Check the list of allowed templates in the parent entity, the classification code of the parent entity and the template. If the incorrect template has been used or if the client is attempting to put the entity in the incorrect place in the classification scheme, the error must first be fixed on the client. Now try saving the object again. Otherwise the template must be added to the list of allowed templates.

 

Frequent problem 3

When attempting to save a new object in IMiS®/ARChive Server from an IMiS®/Scan client, the system returns the message “Error 14”. When attempting to save the object, other clients (IMiS®/Storage Connector, for example) receive the following message:

```
Not enough space on storage profile »%PROFILE_NAME%«. Required %FILE_SIZE% bytes..
```

The server is accessible on the network, and the service at the ports where it is listening is responding (checking using the telnet program):

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=3 ttl=64 time=0.186 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=4 ttl=64 time=0.183 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=5 ttl=64 time=0.164 ms


--- iarc.acme.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.164/0.450/1.530/0.540 ms

[user1@test ~]# telnet iarc.acme.com 16807
Trying 10.1.1.10...
Connected to iarc.acme.com.
Escape character is '^]'.
Connection closed by foreign host.
[user1@test ~]#
```

The server is running. This can be checked using a console command on the server where it is installed:

```
[user1@iarc ~]# sudo service iarcd status
Status of IMiS/ARChive HSM Storage Server: iarcd (pid 23209 23203) is running...
[user1@iarc ~]#
```

Cause of problem 3

All volumes within the saving profile used when saving the new object are full.

Solution for problem 3

Add an adequate number of new volumes to the server profile that does not have sufficient space.

 

Frequent problem 4

When starting IMiS®/ARChive Server the following message appears on the console:

```
[user1@iarc ~]# sudo service iarcd start

WARNING: Network subsystem not running or (RT)NETLINK interface not configured in this kernel. If you're sure
that your network is UP you can ignore this message. Continue loading IMiS/ARChive HSM Storage Server...

Starting IMiS/ARChive HSM Storage Server:                [  OK  ]
[user1@iarc ~]#
```

The server is not accessible on the network:

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
... (pause) ...
From 192.168.92.32 icmp_seq=2 Destination Host Unreachable
From 192.168.92.32 icmp_seq=3 Destination Host Unreachable
From 192.168.92.32 icmp_seq=4 Destination Host Unreachable
... (stop test using CTRL-C) ...
^C
--- iarc.acme.com ping statistics ---
7 packets transmitted, 0 received, +3 errors, 100% packet loss, time 6937ms
[user1@test ~]#
```

Cause of problem 4

When starting up the server, the network subsystem of the operating system was not working.

Solution for problem 4

Get the network subsystem running and then try to start the server again. If the message appears again, the server is probably incompatible with the operating system.

Frequent problem 5

When starting IMiS®/ARChive Server the following message appears on the console:

```
[user1@iarc ~]# sudo service iarcd start

WARNING: Network subsystem not running or (RT)NETLINK interface not configured in this kernel. If you're sure
that your network is UP you can ignore this message. Continue loading IMiS/ARChive HSM Storage Server...

Starting IMiS/ARChive HSM Storage Server:                [  OK  ]
[user1@iarc ~]#
```

 

The server is not accessible on the network:

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
... (pause) ...
From 192.168.92.32 icmp_seq=2 Destination Host Unreachable
From 192.168.92.32 icmp_seq=3 Destination Host Unreachable
From 192.168.92.32 icmp_seq=4 Destination Host Unreachable
... (stop test using CTRL-C) ...
^C

--- iarc.acme.com ping statistics ---
7 packets transmitted, 0 received, +3 errors, 100% packet loss, time 6937ms
[user1@test ~]#
```

The following records appear in this order in the log:

<date and time of record> [iarcd:<decimal value>:<decimal value>] INFO[6] Preforking 1 connection handling childs.
<date and time of record> [iarcd:<decimal value>:<decimal value>] WARN[4] Cannot bind socket 0 to address [10.1.1.10] on port [16807], error 99: Cannot assign requested address. Socket will be closed.
<date and time of record> [iarcd:<decimal value>:<decimal value>] ERR[3] Server was unable to open any configured listening socket.
<date and time of record> [iarcd:<decimal value>:<decimal value>] INFO[6] Child 2922 exited with exit code 0.
<date and time of record> [iarcd:<decimal value>:<decimal value>] INFO[6] Fatal error occured. Server is shutting down.

Cause of problem 5

When starting the server, the network subsystem of the operating system was not running or the incorrect network settings were set in the configuration file.

Solution for problem 5

Check the operation of the network subsystem and change the network settings in the configuration file `/etc/iarc.conf` accordingly.

Frequent problem 6

When starting IMiS®/ARChive Server the following message appears on the console:

```
[user1@iarc ~]# sudo service iarcd start
Error accessing IMiS/ARChive Database directory (<path-to-database>). Check user iarc access to this directory (must be rwx)
[user1@iarc ~]#
```

 

The server is otherwise available on the network:

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=3 ttl=64 time=0.186 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=4 ttl=64 time=0.183 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=5 ttl=64 time=0.164 ms

--- iarc.acme.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.164/0.450/1.530/0.540 ms
[user1@test ~]#
```

Cause of problem 6

The server execution program cannot access its own internal database due to:

- Incorrect settings in the /etc/iadbprovider.xml configuration file, in the /IMiSARChive/Configuration/Database/DriverArguments/DatabaseLocation section and/or incorrectly configured access rights and/or ownership of set directory.
- The internal database cannot be accessed because the disk on which the database is located is not connected to the correct directory.

Solution for problem 6

Check that the settings for the location of the internal server database are correct (in /etc/iadbprovider.xml, the section /IMiSARChive/Configuration/Database/DriverArguments/DatabaseLocation, if this section exists. If this setting is not in the configuration file, the default setting is /iarc/db. Check the rights and ownership of the directory and the files inside the directory listed as the directory that contains the internal database files. The user performing server processes (by default iarc ) must have rights to read, write and create new files in this directory. The group to which the user belongs (by default iarc) only needs read rights. If the /iarc directory is empty, this is probably because the disk where the internal server database is located in the directory (default /iarc/db) cannot be accessed; in this case, the disk must first be accessible.

 

Frequent problem 7

After a complete reboot of IMiS®/ARChive Server, attempts to view objects stored on the server in IMiS®/Scan and IMiS®/View return "Error 61523". Other clients (IMiS®/Storage Connector, for example) return the following error:

IMiS/ARC Client <IASession.Open> Failed to establish connection to the cluster node <10.1.1.10, 16807> (Reason: Error <TimedOut> occurred while opening network connection.).

The server is otherwise available on the network:

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=3 ttl=64 time=0.186 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=4 ttl=64 time=0.183 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=5 ttl=64 time=0.164 ms

--- iarc.acme.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.164/0.450/1.530/0.540 ms

[user1@test ~]#
```

The server is otherwise running, but the status of the server as a service returns the following:

```
[user1@iarc ~]# sudo service iarcd status
Status of IMiS/ARChive HSM Storage Server: iarcd is stopped
[user1@iarc ~]#
```

Cause of problem 7

The initialization script is not activated in the sequence for starting server services.

Solution for problem 7

Automatic start up of the server as a service when the operating system starts up must be changed:

```
[user1@iarc ~]# sudo chkconfig iarcd on
[user1@iarc ~]#
```

Check if the command was successfully executed:

```
[user1@iarc ~]# sudo chkconfig iarcd --list
iarcd          0:off   1:off   2:on    3:on    4:on    5:on    6:off [user1@iarc ~]#
```

 

Then start the server with the following command:

```
[user1@iarc ~]# sudo service iarcd start
Starting IMiS/ARChive HSM Storage Server:          [  OK  ]
[user1@iarc ~]#
```

Frequent problem 8

When starting IMiS®/ARChive Server the following message appears on the console:

```
[user1@iarc ~]# sudo service iarcd start
Starting IMiS/ARChive HSM Storage Server:          [  OK  ]
WARNING: Maximum number of file handles (ulimit -n) allowed for
user iarc or group iarc is 1024. Set allowable maximum to
at least 4096 by adding following two lines to /etc/security/limits.conf:
iarc        hard        nofile      4096
iarc        soft        nofile      4096
 or
@iarc       hard        nofile      4096
@iarc       soft        nofile      4096
If you still recieve this message after modifying /etc/security/limits.conf
check if Pluggable Authentication Modules (PAM) include module
pam_limits.so in session service for user iarc and/or group iarc
(see Linux-PAM system administrators guide on how to manage modules)
IMiS/ARChive will continue to run normally with current setting...
                                        [  OK  ]
[user1@iarc ~]#
```

After start up the service will run normally. Over time, the server becomes inaccessible to the sessions of new clients. The following records appear in the log:

```
<date and time of enty> [iarcd:<decimal value>:<decimal value>] CRIT[2] No child process can accept new
connection.
```

Cause of problem 8

The server has reached the maximum number of open files and cannot accept new connections. The operating system recognizes each connection as an “open file”.

Solution for problem 8

Check the system setting for the maximum number of open files for the iarc user for whom the server is running.

 

## 9.3 Less frequent problems

Less frequent problem 1

When attempting to view an object saved on IMiS®/ARChive Server, IMiS®/Scan or IMiS®/View clients return “Error 11”.

Other clients (IMiS®/Storage Connector, for example) receive the following message when attempting to retrieve the object from the server:

```
iavol OpenObject error 0x564e4f56..
```

The server is accessible on the network, and the service at the port where it is listening is responding (checked using the telnet program).

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=3 ttl=64 time=0.186 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=4 ttl=64 time=0.183 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=5 ttl=64 time=0.164 ms

--- iarc.acme.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.164/0.450/1.530/0.540 ms

[user1@test ~]# telnet iarc.acme.com 16807
Trying 10.1.1.10...
Connected to iarc.acme.com.
Escape character is '^]'.
Connection closed by foreign host.
[user1@test ~]#
```

The server is running. This can be checked using a console command on the server where it is installed:

```
[user1@iarc ~]# sudo service iarcd status
Status of IMiS/ARChive HSM Storage Server: iarcd (pid 23209 23203) is running...
[user1@iarc ~]#
```

Cause of problem 1

The client is attempting to open an object that is correctly entered in the internal server database but the content of the object is not where it should be or is missing.

 

Solution for problem 1

The following will be needed:

- Data on the object identifier from the application using the archive system (for example 4c9f36d38b4d6985b1ec111a5a14a7e9db89edd0cb36923010b6624c667ef142);
- The content of the IdentPassword parameter from the server configuration file /etc/iarc.conf.
- Information about the customer.

Send this information to the email address *support@imis.eu*.

Our technical staff will then decrypt the object identifier that is the basis of the information and further restoration processes with backups or searches in the database system if it is not located in its source location. This is only possible if someone with administrator rights on the server has moved or deleted the object from its original location.

Less frequent problem 2

When attempting to view an object saved on IMiS®/ARChive Server, IMiS®/Scan or IMiS®/View clients return "Error reading IMiS object". Other clients (IMiS®/Storage Connector, for example) receive the following message when attempting to retrieve the object from the server:

```
Unable to locate database record for entity <decimal value>
```

The server is accessible on the network, and the service at the port where it is listening is responding (checked using the telnet program)

```
[user1@test ~]# ping iarc.acme.com
PING iarc.acme.com (10.1.1.10) 56(84) bytes of data.
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=1 ttl=64 time=0.653 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=3 ttl=64 time=0.186 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=4 ttl=64 time=0.183 ms
64 bytes from iarc.acme.com (10.1.1.10): icmp_seq=5 ttl=64 time=0.164 ms

--- iarc.acme.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.164/0.450/1.530/0.540 ms

[user1@test ~]# telnet iarc.acme.com 16807
Trying 10.1.1.10...
Connected to iarc.acme.com.
Escape character is '^]'.
Connection closed by foreign host.
[user1@test ~]#
```

 

The server is running. This can be checked using a console command on the server where it is installed:

```
[user1@iarc ~]# sudo service iarcd status
Status of IMiS/ARChive HSM Storage Server: iarcd (pid 23209 23203) is running...
[user1@iarc ~]#
```

Cause of problem 2

The client is attempting to open an object that has not been entered in the internal server database.

Solution for problem 2

The following will be needed:

- Data on the object identifier from the application using the archive system (for example 4c9f36d38b4d6985b1ec111a5a14a7e9db89edd0cb36923010b6624c667ef142).
- The content of the IdentPassword parameter from the server configuration file /etc/iarc.conf.
- The internal database (the content of the /iarc/db directory) in compressed or text form.
- Information about the customer.

Send this information to the email address *support@imis.eu*.

Our technical staff will then decrypt the object identifier that is the basis for the information and further processes for determining the status of the internal database and the reason the record that is the basis for correct operations with the object in the server inventory cannot be found.

## 9.4 List of service errors entered in the operation log

### 9.4.1 Level 0 – Emergency

EMERG: "Unknown exception caught."
EMERG: Exception caught <description>
This error occurs when a serious error occurs in the operations of IMiS®/ARChive Server which is listed or foreseen as a possible error.
There can be a number of reasons for this, from the environment to possible errors in the application code of the service.

 

### 9.4.2 Level 1 – Alert

ALERT: "Out of memory."

This error means that IMiS®/ARChive Server has run out of available working memory.

The server can continue its work in this state, but it is critical. If the service is exposed to this environment for a longer period of time, it could stop working.

ALERT: "Thread <thread identifier> error number <error number>. Exiting..."

This error occurs when an error that cannot be fixed occurs in one of the threads.
Depending on the seriousness of the error, IMiS®/ARChive Serverv can terminate the thread or the entire program executing it, as continuing could threaten the consistency of the persistent data.

ALERT: "Maximum number of child processes reached."

This notification means that IMiS®/ARChive Server can no longer start its connecting sub-process and therefore cannot process new requests.
The server will continue running until new resources become available or the number of sub-processes is increased (setting in the /etc/iarc.conf file).

ALERT: "Shared memory (hnd = <no.>, ptr = <no.>) error <error number>"

This notification appears if IMiS®/ARChive Server has detected an error or impropriety in work with a portion of shared memory used for communication between processes. The server stops immediately because communications between processes in the IMiS®/ARChive family are no longer possible.

### 9.4.3 Level 2 – Critical

ALERT: "Out of memory. Cannot continue."

This error means that IMiS®/ARChive Server has run out of available working memory.
The server can continue its work in this state, but it is critical. If the service is exposed to this environment for a longer period of time, it could stop working.

CRIT: "Error is unrecoverable. The process will terminate."

An error has occurred due to which the process cannot continue working.
This notification is usually only the result of some other error that occurred immediately prior to this and that is also listed in the error log.

CRIT: "Unsupported client address structure at <process number>."

IMiS®/ARChive Server has detected an unsupported client address structure in process <no. of process>. The error means communication between the client and server has been terminated.

 

CRIT: "Signal SIGSEGV occured Process will shut down ..."
An error occured in work with IMiS®/ARChive Server's working memory.
The server immediately shuts down. The server administrator will need to intervene, and will usually have to restart the process.

CRIT: "No child process can accept new connection."
Communication sub-processes cannot accept new requests.
This error is usually the result of another error or a lack of server resources.

CRIT: "Cannot initialize LC_COLLATE setting."
CRIT: "Cannot initialize LC_CTYPE setting."
IMiS®/ARChive Server cannot set the desired region settings for sorting and translating characters outside the scope of the ACSII code table.
If this setting is just one of the available settings when the `locale –a` command is run, this error is only theoretical.

CRIT: "New process couldn't accept new connection."
IMiS®/ARChive Server ran out of resources. It started a new communication process but there aren't enough resources to accept the new requests.

CRIT: "Error <error number> while recording session close to audit log."
CRIT: "Error < error number > while recording session open to audit log."
An error occurred when making an entry in the audit log, the entry was not successfully sent and entered in the internal database. A number of causes are possible. The server administrator will need to intervene.

CRIT: "Locale parameter '<parameter value>' for LC_COLLATE in not formated according to POSIX standard and cannot be used. Use format ll_CC[.CHARSET[@variant]].«
CRIT: "Locale parameter '<parameter value>' for LC_CTYPE in not formated according to POSIX standard and cannot be used. Use format ll_CC[.CHARSET[@variant]]."
The given regional settings parameter for sorting and translating characters outside the scope of the ASCII code table is not in the correct POSIX format.
See the `CountryLanguage` setting in the `[Server]` section of the configuration file.

CRIT: "Error <error number> while recording server's session to audit log."
Error entering the server session in the audit log of internal database sessions.
A number of causes are possible. The server administrator will need to intervene.
This error is only theoretical.

 

CRIT: “Error <error number> opening file <file name>. Terminating process.”
CRIT: “Error <error number> while reading file <file name>. Terminating process.”
CRIT: “File <file name> is too small (<number> bytes). Terminating process.”
CRIT: »Invalid header data in file <file name>. Terminating process."CRIT: "Error <error number> seeking in file <file name>. Terminating process.”
CRIT: “File <file name> is too large. Terminating process."CRIT: "File <file name> has invalid size (<number> bytes). Terminating process.”
CRIT: “Error <error number> while reading file <file name>. Terminating process.”
CRIT: “Audit log config check: Crypt engine error. Terminating process.”
CRIT: “Audit log config check: Data size doesn't match. Terminating process.”
CRIT: “Audit log config check: Checksum error. Terminating process.”
CRIT: “Audit log config check: Crypt engine error. Terminating process."CRIT: "Error <error number> while writing to file <file name>. Terminating process.”
CRIT: “Audit log config check: Database error. Terminating process."CRIT: "Audit log config check: Unexpected unknown error. Terminating process.”
The errors listed above indicate problems in the system for detecting changes to the audit log settings. These errors can be resolved by deleting the /iarc/db/iaalcc.bin file; however, they should be taken seriously and their causes should be looked into. Removing this file will cause “changes” to be entered in the audit trail when the server is restarted, even though changes have not occurred.
The current settings will be marked as changed, as the setting status is not available to the service prior to start up (the changes are saved in encrypted binary form in this file).
CRIT: “Error while recording server's session to audit log.”
CRIT: “Malformed Entity identification configuration. Reason: ‘<reason>'”

### 9.4.4 Level 3 – Error

ERR: “Read returned with error: <error number>.”
An error occurred when reading from the network socket for communication with the client. The error is not critical and usually only means a sudden, irregular termination of the session by the client.

ERR: “Write returned with error: <error number>.”
An error occurred when writing for the network socket for communicating with the client. The error is not critical and usually only means a sudden, irregular termination of the session by the client.

ERR: “No select file descriptor available.”
The maximum number of open files has been reached. It is not possible to use the new node (i-node) that IMiS®/ARChive Server needs for work with the object.

 

ERR: “IDFromIdentShort: Unknown ObjectID version information.”
The client requested an object whose identifier structure is not recognized by IMiS®/ARChive Server, or which it cannot decrypt or does not have entered in its internal database.
The server denied the client request.

ERR: “Cannot create object file <file code/name>.”
When entering an object in the volume an error occurred because a file with the object already exists. A specialist from the manufacturer needs to look at it and figure out why a file with the object identifier already exists at this location.

ERR: “Unknown file handling error.”
An unforseen error occurred during work with the object. IMiS®/ARChive Server denied the request.

ERR: “Cannot open object file <object file name>.”
The client has requested an object from IMiS®/ARChive Server that is entered in the inventory, but the object file does not exist.

ERR: “ObjRemove error <error number>.”
IMiS®/ARChive Server has received a regular request to delete an object but deletion cannot be performed. A specialist from the manufacturer needs to have a look to figure out why deletion is not possible (this is usually due to rights over the file system and HSM inventory files).

ERR: “Cannot open object file <object file name>.”
An error occurred when attempting to read an object, IMiS®/ARChive Server cannot open the object. A specialist from the manufacturer needs to have a look because the object is entered in the inventory but the process cannot be performed (this is usually due to rights over the file system and HSM inventory files).

ERR: “Not enough space available in profile <profile ID>.”
There is not enough space in the volumes belonging to a profile <profile ID>.

ERR: “Invalid profile number.”
An incorrect or non-existent profile number was used in the client’s request.

ERR: “Unexpected FIN!”
An IMiS® client unexpectedly closed a session or sent a signal to close a session after IMiS®/ARChive Server already closed the session because it was inactive.

 

ERR: "Error in ConnInfoGetLib request (req->seq). Skipping processing."
An IMiS® client sent an incorrect request for the communication library to IMiS®/ARChive Server.

ERR: "Cannot open file <library name>."
An IMiS® client sent IMiS®/ARChive Server the correct request for a communication library, but the library is not where it should be. This error is usually the result of incomplete installation or incorrect installation in the /etc/iarc.conf configuration file or a problem with the rights of the iarc user.

ERR: "Unknown object handle <number/handle>."
The operating system on IMiS®/ARChive Server sent an irregular object file handle.
This error is most likely the result of the operating system or file system not functioning properly.

ERR: "Unknown transmission handle."
The operating system on IMiS®/ARChive Server sent an irregular handle for the connection node (i-node). This error is most likely the result of the not functioning properly.

ERR: "Unknown ConnInfo request: <number> - ignoring!"
IMiS®/ARChive Server received an irregular request for data about the connection to the client. The server rejected the request.

ERR: "Unknown External ID request size (request size)."
IMiS®/ARChive Server received a request for an identification number for a new object for a so-called external system (SAP/R3 for example). The size of the requested identification number is not regular. The server rejected the request as invalid.

ERR: "Invalid request size: <request size>."
IMiS®/ARChive Server received a request with an irregular size. The server rejected the request as invalid.

ERR: "Unknown request <request number> received. Closing connection."
IMiS®/ARChive Server received an irregular request and closed the open connection.
This is usually the result of attempting to connect through the server's TCP port with a protocol that the server does not recognize.

ERR: "Socket <socket number> closed for reading on client side. Connection closed."
IMiS®/ARChive Server detected that the socket for communicating with the client is closed, which is why it also closed the connection on its end.
ERR: "Socket <socket number> write error <error number>.«

 

IMiS®/ARChive Server cannot communicate with the client through the socket.

ERR: “Error reading message queue (error: <error code>).”
When exchanging data between processes in the IMiS®/ARChive Server family, an error occurred while reading data from the queue for inter-process communication.

ERR: “Ident(): Initializing crypto engine.”
An error occurred when initializing the system for object identifiers encryption.
This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

ERR: “Ident(): Setting internal key.”
An error occurred when setting a key for the first level of object identifier encryption.
This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

ERR: “Ident(): Setting external key.”
An error occurred when setting a key for the second level of object identifier encryption.
This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

ERR: “Ident(): Internal encrypting.”
An error occurred on the first level of object identifier encryprion.
This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

ERR: “Ident(): External encrypting.”
An error occurred on the second level of object identifier encryption. This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

ERR: “IDFromIdent(): Initializing crypto engine.”
An error occurred when initializing the system for object identifiers decryption.
This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

 

ERR: "IDFromIdent(): Setting external key."
An error occurred when setting a key for the first level of object identifiers decryption.
This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

ERR: "IDFromIdent(): Setting internal key."
An error occurred when setting a key for the second level of object identifier decryption.
This error is only theoretical and is the result of an error by the programmer; a specialist from the manufacturer should be consulted.

ERR: "IDFromIdent(): External decrypting."
An error occurred on the first level of object identifier decryption. This error can occur because an incorrect identifier was sent by the client.

ERR: "IDFromIdent(): Internal decrypting."
An error occurred on the second level of object identifier decryption. This error can occur because an incorrect identifier was sent by the client.

ERR: "IdentShort(): error <error number>while ciphering internal block."
An error occurred on the first level of ciphering a short object identifier.
This error is only theoretical; a specialist from the manufacturer should be consulted.

ERR: "IdentShort(): error <error code>while ciphering external block."
An error occurred on the second level of ciphering a short object identifier.
This error is only theoretical; a specialist from the manufacturer should be consulted.

ERR: "IDFromIdentShort: 1st Server id (<server identifier>) does not match.«
The client sent the incorrect value for the short form of the object identifier.

ERR: "IDFromIdentShort: 2nd Server id (<server identifier>) does not match."
The client sent the incorrect value for the short form of the object identifier.

ERR: "IDFromIdentShort: Unknown ObjectID version information (<decimal value>)."
The client sent the incorrect value for the short form of the object identifier, or the value was created with a newer version of IMiS®/ARChive Server and cannot be deciphered by the server.

 

ERR: "IDFromIdentShort(): error <error number> while deciphering external block."
ERR: "IDFromBytesL(): error <error number> while deciphering external block."
An error occurred on the first level of deciphering the short form of the object identifier. The error occurred because the incorrect identifier was used for the short form sent from the client.

ERR: "IDFromIdentShort(): error <error number> while deciphering internal block."
An error occurred on the second level of deciphering the short form of the object identifier. The error occurred because the incorrect identifier was used for the short form sent from the client.

ERR: "IDFromIdentShort: Invalid ID data."
An error occurred due to incorrect checking data content after deciphering the short form of the object identifier. The error occurred because the incorrect identifier was used for the short form sent from the client.

ERR: "Volume client error."
An unidentified error occurred in the operations of the module for managing disk media. The cause of the error is usually operating system malfunction due to a lack of system resources.

ERR: "Invalid audit query size <decimal value>"
The server received a request for an audit trail query, and the size of the request is not correct. This error is caused by a client that is not functioning properly; it is only theoretical.

ERR: "Invalid sess_cond.type (<decimal value>)"
The server received a request for an audit trail query in which an incorrect value for specifying session criteria was entered. This error is caused by a client that is not functioning properly; it is only theoretical.

ERR: "Invalid sess_cond.offset (<decimal value>)"
The server received a request for an audit trail query in which an incorrect condition structure for searching sessions was entered. This error is caused by a client that is not functioning properly; it is only theoretical.

ERR: "Invalid ts_cond.offset (<decimal value>)"
The server received a request for an audit trail query in which an incorrect condition structure was entered for the event time period. This error is caused by a client that is not functioning properly; it is only theoretical.

ERR: "Invalid objid_cond.offset (<decimal value>)"

 

The server received a request for an audit trail query in which an incorrect condition structure was entered for specifying object identifiers. This error is caused by a client that is not functioning properly; it is only theoretical.

ERR: "AuditQuery::GetNextAddress(), line <decimal value>, error <decimal value>"
The server received a request for an audit trail query in which the conditions for searching the network addresses of clients contain an incorrect form of data. This error is caused by a client that is not functioning properly; it is only theoretical.

ERR: "ObjectsQueryArray::FindEvents(); Error decrypting object id."
The server received a request for an audit trail query in which the object identifier contains incorrect values. This error is caused by a client that is not functioning properly; it is only theoretical.

ERR: "msgctl(<system identifier>, IPC_RMID) error <error number>: <description of system error>"
A system error occurred when setting up the queue for communication between processes, specifically when attempting to remove the current queue. The cause of the error is usually an operating system that is not functioning properly, and it is described in detail in <description of system error>.

ERR: "Could not connect to iavol server. Session canceled."
An error occurred when connecting to the module for work with disk media.
The cause of the error is usually a lack of system resources or the inability of the operating system to serve any more client sessions.

ERR: "BuildVolTree(): Volume <volume ID> not mounted."
An error occurred in the use of a specific volume. The volume cannot be used due to a previous error which has already been entered in the log.

ERR: "Error closing file descriptor."
A system error occurred when closing a file. The error is usually the result of an error by a programmer; it can also be caused by an operating system that is not functioning properly.

ERR: "accept() returned error <error number>."
ERR: "accept() error <error number>: <description of system error>."
An error occurred when starting a new session with a client. The cause of the error is probably an excessive number of currently running client sessions; the second form of the message also contains a description of the cause in <description of system error>.
ERR: "Passed fd <decimal value> is not a listen socket."

 

An error occurred when listening for requests to start new sessions with clients. The error is caused by an error in the program and is only theoretical. An specialist from the manufacturer should be contacted.

ERR: "Error creating thread: Ox<hexidecimal value>."
An error occurred when starting a new process thread in the program. The cause of the error is usually a lack of system resources.

ERR: "Stats counter error <error number>: "<description of system error>."
An error occurred when saving statistical data about the number of instances of access to the objects in a set time period. The error is caused by an operating system that is not functioning properly.

ERR: "Profile(<hexidecimal value >): No volumes on level <level number>. Emergency migration skipped."
ERR: "Profile(<hexidecimal value >): No volumes on level <level number>. Scheduled migration skipped."
An error occurred when attempting to migrate profile objects to a higher level (<level number>) in the volume hierarchy. The error occurred because the profile on this level does not have volumes defined and there is not enough space on the lower level.

ERR: "mkstemp("<file name>") error <error number>."
An error occurred when creating a temporary file with the name <file name>.
The error is the result of insufficient space on the disk and the rights of user iarc; it can also be caused an operating system that is not functioning properly.

ERR: "unlink("<file name>") error <error number>."
An error occurred when deleting a file with the name <file name>.
The error is caused by an error in the program; it is probably an error with the access rights of user iarc or the operating system.

ERR: "User '<user>' at connection <number> denied due to authentication failure - Authentication request sequence mismatch (request subid:proto:stage = COPN_CLIAUTH:SRP6A:< number >, valid COPN_CLIAUTH:SRP6A:< number >, session stage = <number>);"
ERR: "Error creating server SRPC-6a evidence for user '<user>'. Connection <number> denied (reason: <reason>)."

 

ERR: “Connection <number> denied due to authentication failure - Authentication request sequence mismatch (request subid:proto:stage = COPN_CLIAUTH:SRP6A:<number>, valid COPN_CLIAUTH:SRP6A:[<number>/<number>], session stage = <number>);”
ERR: “Connection <number> denied (reason: Unknown Authentication mode requested (id = <number>))”
An error occurred during user authentication. There are a number of possible causes, from incorrect user credentials to incorrect use of the authentication protocol (in rare cases). Access is denied to these sessions.

ERR: “Level of volume in profile exceeds maximal allowed value.”
The current volume level exceeds the maximum allowed value (16).

ERR: “GetVolumeInfo error <error description>.”
An error occurred when opening a service volume in content storage.

ERR: “"Error connecting to database. Reason: ‘<error description>' at’<file>:<row>'”
An error occurred when connecting to the database. If this error occurs a specialist from the manufacturer should perform a database check.

ERR: “Error opening archive.”
An error occurred when opening the archive. This is probably due to an error in the data or inconsistent archive settings. If this error occurs a specialist from the manufacturer should perform a check up.

ERR: “IAVolume: Unable to retrieve a database session!”
The program is having problems retrieving a database session. If this error occurs a specialist from the manufacturer should perform a check up.

ERR: “Database exception <error description>!”
The program is having problems accessing the database. The solution depends on the error description. If a solution cannot be found based on the description, a specialist from the manufacturer should be consulted.

ERR: “Access mode ‘<access mode>' is unsupported.”
ERR: “Unsupported access mode ‘<access mode>'.”
An invalid access mode is listed in a request. The allowed values are: “RO” and “RW”.
This is an error on IMiS®/Client.

ERR: “ACL entry for ‘<directory entity>' is internal and cannot be updated.”
The access control list cannot be changed for system directory entities.
ERR: “Address '< network address>' structure is not supported.”

 

The incorrect network address format has been used. A valid host name, IPv4 or IPv6 address must be entered with an optional extension for the network port. The network address and port are separated with a colon, ":" .
This is an error in the product settings.

ERR: "Attribute 'destination' contains the value '%s' which resolves to an unknown attribute"
ERR: "Attribute 'destination' identifies the attribute '%s' which type '%u' is not supported for entity identification storage"
An error occurred when setting the counters for automatic entity numbering.

ERR: "AuditQuery::GetNextAddress(), line %d, error %d"
Error getting the IP address from the request to view the audit trail.

ERR: "Binary, File and StringMax are unsortable."
Sorting of attributes of the types "Binary", "File" and "StringMax" is not possible.
The sorting function is not being used properly on an IMiS®/Client.

ERR: "Both IP addresses need to be of the same length and same protocol."
Error getting the range of IP addresses from a request to view the audit trail.
Both end values of the range must be from the same family. IMiS®/Client has sent an incorrect range of values.

ERR: "Can't move an entity into its own subtree."
An entity cannot be moved into its own child entity.

ERR: "Classification code '<classification code>' exceeds maximum length of 20 characters."
The largest allowed length of a classification code is 20 characters.

ERR: "Collection handle '<hexidecimal value>' is invalid."
The reference (handle) to the collection is not valid. Possible reasons for this error include:

- The collection was previously closed.
- The session from which the reference originates was closed and then reopened as a new session.
- The handle never existed.

This is an error on IMiS®/Client.

 

ERR: “Configuration requires client computer name to be provided.”
IMiS®/ARChive Server is set up so that when a new session is opened it requires the name of the computer from which the session was started.
This is a settings error if the name of the computer is not required or an error in IMiS®/Client, if the name of the computer is required.

ERR: “Configuration requires client username to be provided.”
IMiS®/ARChive Server is set up so that when opening a new session it demands the name of the user who started a session with authentication methods before SRP-6a.
This is a settings error if the user name shouldn’t be required information or a client error if the user name is required.

ERR: “Counter definition ‘<counter definition >' does not contain a required variable ‘<variable name>'.”
An error occurred in the counter definition. The name of the variable could not be found in the definition of the counter for automatically creating classification codes.

ERR: “Decrypted Authentication payload starting sequence is invalid.”
This is an authentication error. This is an error on IMiS®/Client.

ERR: “Destination is already a parent of the specified entity. Move is not possible.”
The move would not change the location of the entity in the classification scheme and therefore does not make sense. The move is not performed. This is an error on IMiS®/Client.

ERR: “Element count must be greater than 0.”
The number of elements on the side of the entity collection must be positive.
This is an error on IMiS®/Client.

ERR: “Empty request received.”
A request with no content was received. This is an error on IMiS®/Client.

ERR: “Encoding (<number>) not recognized.”
The requested encoding of unique entity identifiers in the audit query results is not recognized. The following values are allowed: 1 (BASE 16), 2 (BASE 64) and 3 (BASE 85).

 

ERR: “Entity handle ‘<hexidecimal number>is invalid.”
ERR: “Entity handle is invalid.”
The reference (handle) to the entity is not valid. Possible reasons for this error include:

- The entity was previously closed.
- The session from which the reference originates was closed and then reopened as a new session.
- The reference never existed.

This is an error on IMiS®/Client.

ERR: “Entity handle is invalid or you're trying to open a binary object in read-write mode while the parent entity is opened in read-only mode.”
The reference is invalid (for the above error description *see error Entity handle is invalid*)
or an attempt has been made to open a binary object in Read and Write mode although the parent entity is open in Read Only mode.

ERR: “Entity handle must be provided.”
A request without a handle is invalid. This is an error on IMiS®/Client.
ERR: “Entity id type cannot be NONE.”
The unique entity identifier type in the request cannot be N (NONE).
The following values are allowed: I (encrypted internal), E (external) or C (classification code).
This is an error on IMiS®/Client.

ERR: “Entity identifier not specified.”
ERR: “Entity unique identifier (id or handle) must be provided.”
The request does not contain a valid unique entity identifier.
This is an error on IMiS®/Client.

ERR: “Entity type ‘<type>' is unknown.”
The request does not contain a valid entity type. The following values are allowed: C (Class), F (Folder) and D (Document). This is an error on IMiS®/Client.

ERR: “Entity type ‘<number>' is not a valid template entity type.”
ERR: “Template enumeration type ‘<number>' is not a valid entity type.”
The template contains an invalid value for the type of entity it describes.
A range of values from 1 to 5 is allowed. This is an error on IMiS®/Client.

ERR: “Error converting IP address”
An error occurred when converting a character string to an IP address.
This is an error on IMiS®/Client.

 

ERR: “Exactly 2 id tags are required in mv request.”
A request to move must contain exactly two unique entity identifiers.
This is an error on IMiS®/Client.

ERR: “Insufficient rights to create file property in the specified enity.”
ERR: “Insufficient rights to create subentities in the destination entity.”
ERR: “Insufficient rights to create subentities under the specified parent.”
The user does not have sufficient rights to perform these operations. If the user should have the rights needed to perform these operations, the Access Control List should be checked and updated as needed.

ERR: “Invalid acl scope request ‘<character>'.”
An invalid identifier for the Access Control List elements has been used in a request.
The following values are allowed: N (None), E (entities) and A (Attributes).
This is an error on IMiS®/Client.

ERR: “Invalid date part (<date part>).”
ERR: “Invalid date part.”
ERR: “Invalid time part <time part>).”
ERR: “Invalid time part.”
ERR: “Invalid timestamp.”
An error occurred when converting a character string to a date/time.

ERR: “Invalid old password.”
ERR: “New password required.”
ERR: “Old password required.”
An error has occurred when attempting to change a password. Either the old password has not been given (old password required), or is incorrect (invalid old password) or the new password has not been entered (new password required). This is an error on IMiS®/Client.

ERR: “Invalid property scope ‘<character>' requested.”
An invalid value was entered for the scope of returned properties in an “rd” request.
The following values are valid: N (No return), S (Listed), A (All) in P (Public).
This is an error on IMiS®/Client.

ERR: “Invalid tag %u.”
ERR: “Invalid tag name %u.”
An error occurred executing the request. An invalid XML tag was used.
This is an error on IMiS®/Client.

ERR: “Invalid value id ‘<number>' cannot be opened.”

 

An invalid value was used for a binary content identifier.
This is an error on IMiS®/Client.

ERR: “Invalid XML request: Unknown root tag ‘<name>' found.”
An unknown root tag was used in the request.
This is an error on IMiS®/Client.

ERR: “Invalid/Unsupported key type <identifier>.”
An unsupported type of authentication key was used.
This is an error on IMiS®/Client.

ERR: “Malformed classification code ‘<code>' (<error>).”
A malformed classification code was used.

ERR: “Malformed counter definition '%s'.”
A malformed definition was used for the counter for generating classification codes.
This is an error on IMiS®/Client.

ERR: “Malformed expression '%s'.”
A malformed expression was used in a codelist.
This is a configuration error.

ERR: “Newly created Binary object identified by '%llu' was not created from this session and cannot be assigned to the property '%s'.”
ERR: “Newly created Binary object identified by '%llu' was not created from this session.”
An invalid binary object identifier was sent in the request.
This is an error on IMiS®/Client.

ERR: “No legacy archival configuration for profile '%s'”
The archive profile is not included in the configuration for archiving.

ERR: “Page element count ‘<number of elements>' exceeds maximum allowed size of '< number of elements>' elements.«
The number of elements of the page with the entity collection must be smaller than the largest allowed number of elements (10000). This is an error on IMiS®/Client.

ERR: “Parent required but not specified.”
The request must contain the unique identifier of the parent entity under which the new entity is being created. This is an error on IMiS®/Client.
ERR: “Property ‘<attribute name>' is not a streamable property.”

 

An incorrect attribute type was used. The data stream is only possible with attributes of the types “Binary”, “File” and “StringMax”. This is an error on IMiS®/Client.

ERR: “Property code ‘<attribute name>': Malformed boolean value ‘<value>'.”
An incorrect value was given for a binary attribute in the request. The following values are allowed: 0, 1, true or false. This is an error on IMiS®/Client.

ERR: “Property name must be provided.”
The request must contain a name of the attribute. This is an error on IMiS®/Client.

ERR: “Provided client application name contains invalid UTF-16 characters.”
The name of the client in the request contains a UTF-16 character that is not allowed.
This is an error on IMiS®/Client.

ERR: “Provided client application name exceeds maximum length of <number> UTF-16 characters.”
The client name in the request is too long. This is an error on IMiS®/Client.

ERR: “Provided client local address '<IP address>' cannot be converted into its binary form (Detail: <error description>)”
An error occurred when converting an internal IP address. This usually means that the wrong form of the address was sent. This is an error on IMiS®/Client.

ERR: “Provided client local address contains invalid UTF-8 characters.”
The client’s local IP address in the request contains a UTF-8 character that is not allowed.
This is an error on IMiS®/Client.

ERR: “Provided client local address exceeds maximum length of <number> UTF-8 characters.”
The client’s IP address in the request is too long.
This is an error on IMiS®/Client.

ERR: “Provided computer name contains invalid UTF-16 characters.”
The name of the user’s computer in the request contains a UTF-16 character that is not allowed. This is an error on IMiS®/Client.

ERR: “Provided computer name exceeds maximum length of <number> UTF-16 characters.”
The name of the user’s computer in the request is too long.
This is an error on IMiS®/Client.

ERR: “Provided username contains invalid UTF-16 characters.”

 

The user name in the request contains a UTF-16 character that is not allowed.
This is an error on IMiS®/Client.

ERR: “Provided username exceeds maximum length of <number> UTF-16 characters.”
The user name in the request is too long. This is an error on IMiS®/Client.

ERR: “Reason for deletion required but missing or empty.”
The user is required to give a reason for deletion. This is an error on IMiS®/Client.

ERR: “Server was unable to decrypt authentication payload using any configured crypto contexts.”
A problem occurred in an older interface, as it was not possible to decrypt the authentication payload. This is an error on IMiS®/Client.

ERR: “Start index ‘<start index>' exceeds collection element count ‘<number of elements>'.”
An invalid start index was used in the collection. The start index must not be greater than the number of elements in the collection. This is an error on IMiS®/Client.

ERR: “Template ‘<name>' is internal and cannot be used.”
Internal templates cannot be used.
This is an error on IMiS®/Client.

ERR: “The password is not encoded in correct UTF-8 sequence.”
The password is not a correct UTF-8 character string.
This is an error on IMiS®/Client.

ERR: “The SRP6A group you provided is not supported (id=<number>).”
The SRP6A group is not supported.

ERR: “Time must be in range!”
ERR: “Timestamp must be a range.”
The request for a query of the audit trail must contain a time range for the search.
This is an error on IMiS®/Client.

ERR: “Unable to make query without any parameters.”
The request for a query of the audit trail must contain at least one parameter.
This is an error on IMiS®/Client.

ERR: “Unable to search objects by object id range.”
ERR: “We cannot have ranges of objects”

 

The request for a query of the audit trail cannot contain a range of unique object identifiers. This is an error on IMiS®/Client.

ERR: “Unknown collection management operation ‘<character>'.”
The request contains a collection management value that is not allowed. The values C (Delete collection) and I are allowed. This is an error on IMiS®/Client.

ERR: “Unknown property ‘<attribute name>'.”
ERR: “Unknown property code ‘<attribute name>'.”
This attribute does not exist. This is an error on IMiS®/Client.

ERR: “Unknown stream handle (<hexidecimal number>). Ignoring request.”
An incorrect reference (handle) to a data stream has been given.
This is an error on IMiS®/Client.

ERR: “Unknown value id ‘<number>' requested or error opening a stream.”
An invalid value was used for a binary content identifier.
This is an error on IMiS®/Client.

ERR: “Unknown XML Parser error.”
An unknown error occurred when parsing an XML request.
This is an error on IMiS®/Client.

ERR: “Unknown xml tag encountered: ‘<number>'.”
ERR: “Unknown xml tag encountered: '<string>'”
When parsing an XML request, an unknown XML tag appeared.
This is an error on IMiS®/Client.

ERR: “Unsupported object id size.”
Only two lengths are allowed for reading unique entity identifiers: 24 and 32 characters.
This is an error on IMiS®/Client.

ERR: “XML Parser error: domain=%d, code=%d, msg='%s', level='%s', line=%d”
A problem occurred when parsing an XML request. This is an error on IMiS®/Client.

ERR: “You cannot assign an existing Binary object to the property '%s' from another entity. The offending value is '%llu'.”
The same binary attribute cannot be assigned to two different entities.
ERR: “You cannot create a <entity type> from template ‘<template name>' (Reason: Template is not of type <entity type>).”
The template cannot be used for the given entity type.

 

This is an error on IMiS®/Client.

ERR: “Unable to open the listening port on address ‘[<address>]:<port>' (Reason: ‘<reason>').”
ERR: “Unable to bind the listening socket to address '[%s]:%u' (Reason: '%s').”
An error occurred when opening the network address for serving requests. The reason is usually listed in the “reason” part of the message.

ERR: “Access denied. System entities cannot be opened.”
The “open” operation cannot be performed on system entities.

ERR: “Classification code cannot be set for existing entities.”
The classification code cannot be changed for an existing entity.

ERR: “Classification code is set to be generated automatically.”
The classification code is automatically generated. It cannot be set manually.

ERR: “Invalid stream object for CS_OBJ_UPDATE request.”
An invalid request has been made to open a stream on an old interface.

ERR: “Only end templates can be removed.”
The template the user would like to remove cannot have child templates.

ERR: “Entites based on template ‘<template name>' exist. Template cannot be removed.”
A template cannot be removed if it was used for at least one entity.

ERR: “Invalid CS_OBJ_CREATE request packet size: <length>.”
ERR: “Invalid CS_OBJ_OPEN request packet size: ‘<length>'.”
ERR: “Invalid CS_OBJ_UPDATE request packet size: ‘<length>'.”
Invalid request length with an old interface.

ERR: »<attribute name>' property not found.”
This attribute does not exist.

ERR: “Default storage profile is not set.”
An error occurred in the settings. A default storage profile must be defined.

ERR: “Storage profile ‘<ID>' doesn't exist.”
An error occurred in the settings. A storage profile with this ID does not exist.

ERR: “Error acquiring an instance of Full Text Index Service provider.”

 

An error occurred in the settings.

ERR: “Classification code is set to be generated automatically but generator is not configured for the entity's hierarchy level.”
An error occurred in the settings. Automatic classification code generation has not been set for all levels in the classification scheme.

ERR: “Classification code must be set by creator and cannot be empty.”
The classification code is required information.

ERR: “Classification code must be set to be generated automatically for the destination entity's subentities.”
ERR: “Classification code must be set to be generated automatically for the whole moving entity's hierarchy.”
ERR: “Missing classification code generator. Entity type: ‘<entity type>', Absolute level: <level>, Relative level: <level>.2
An error occurred in the settings. The move operation requires automatic classification code generation to be set in the target tree.

ERR: “Directory entity with name ‘<name>' (id=<number>) is deleted.”
ERR: “Group ‘<name>' (id=<number>) has been deleted.”
A user is attempting to delete an entity that has already been deleted.

ERR: “Directory entity ‘<name>' (id=<number>) has been deleted while being edited.
Group ‘<name>' (id=<number>) has been deleted while being edited.”
A user is attempting to edit an entity that has already been deleted.

ERR: “Directory entity ‘<name>' (id=<number>) is not being edited.”
ERR: “Group ‘<name>' (id=<number>) is not being edited.”
A user is attempting to edit an entity that was not opened in Write mode.

ERR: “Non-empty entities cannot be deleted.”
An entity being deleted cannot contain other entities. The child entities must first be deleted.

ERR: “Entity '<id>' cannot be opened in exclusive mode.”
The entity is already open in Write mode in some other session.
ERR: “Value '<id>' is currently being edited. Close all editable streams before opening a new one.”
ERR: “Value '<id>' is currently being edited through stream ‘<stream reference>'. Close it before opening a new one.”
An attribute of the data stream type is already open in write mode.

 

ERR: “Closed entities cannot be edited.”
The status of the entity in question (or its parent) is “Closed”. No changes can be made to a closed entity.

ERR: “Access denied. Request requires user-credentials authenticated session or higher.”
To execute the desired operation, a session must be opened with authentication.

ERR: “Adding new entities under closed entities is not allowed.”
The status of the entity in question (or its parent) is “Closed”. New child entities cannot be added under a closed entity.

ERR: “Destination status is 'Closed'. Adding new entities under it is not allowed.”
No changes to an entity with “Closed” status are allowed. This includes adding new child entities.

ERR: “Access denied. (You do not have the right to delete the property ‘<name>')”
The user does not have the right to delete attributes in the entity.

ERR: “Access denied. (You do not have the right to create the property ‘<name>')”
The user does not have the right to add attributes to the entity.

ERR: “Access denied. (You do not have the right to edit the property ‘<name>')”
The user does not have the right to edit attribute values in the entity.

ERR: “Query result set is too large. Try to decrease datetime range.”
The result of the audit trail query includes more results than the system is capable of processing. The search parameters need to be changed to return fewer results.
The easiest way to do this is by setting a smaller time range for the searched events.

ERR: “Malformed entity id”
ERR: “Unable to decrypt entity id”
An error occurred when decrypting a unique entity identifier.
This is an error on IMiS®/Client.

 

### 9.4.5 Level 4 – Warning

WARN: “File descriptor <decimal value> is too big for select(). Just closing.”
A function call occurred for closing a user session with an incorrect parameter.
The cause of this warning is an error in the program. This error is only theoretical,
and a specialist from the manufacturer should be consulted.

WARN: “Object header: Illegal server ID.”
The object was probably moved from a different server or the object file is corrupt. It is also possible that IMiS®/ARChive Server was upgraded using a package with a different server identifier. In all three cases, a specialist from the manufacturer should be consulted.

WARN: “Object header: Illegal head.”
An object file is probably corrupted, or it was replaced with a file with incorrect content outside of IMiS®/ARChive Server. As far as the server is concerned, this file cannot be used or read.

WARN: “Object header: Object ID mismatch (ObjID: < hexidecimal value>; Header: <hexidecimal value>).”
The object has an incorrect object identifier entered in its header. The object could be corrupt, or the error is the result of an error that occurred when the object identifier was defined. A specialist from the manufacturer should be contacted. As far as IMiS®/ARChive Server is concerned, this file cannot be used or read.

WARN: “No available volume found in profile <decimal value>.”
Not enough space on the saving profile. A new volume needs to be added or the existing profile volume must be enlarged.
WARN: “Seek offset underflow. Repositioning.”
An attempt has been made to access an object at an invalid location. This error is caused by the incorrect functioning of IMiS®/Client. This error is only theoretical.

WARN: “Unsupported Cipher algorithm requested for 128bit key strength (alg_id=<decimal value>)", alg_id).”
WARN: “Invalid Cipher mode requested (id=<decimal value>)", mode_id).”
WARN: “Unsupported Cipher key strength requested (key_strength=<decimal value>)", key_strength).”
WARN: “Unsupported block size identifier (<decimal value>)", bs).”
WARN: “Block size identifier (<decimal value>) doesn't match crypto context.", bs).”
WARN: “Crypto exception occured (details: <description of details>)", e.what()).”

 

These warnings appear when an IMiS®/Client requests an encryption method that is not enabled in the server settings. The error is caused by incorrect encryption subsystem settings in the client or by inadequate settings on the server in combination with the client settings.

WARN: "Unknown exception occurred while setting up crypto context."
An unexpected error occurred when setting up a context for encrypted communication with IMiS®/Client. This error is only theoretical. An specialist from the manufacturer should be consulted.

WARN: "Audit Log Query session denied by configuration settings."
IMiS®/ARChive Server denied a request to start a session for viewing the audit log because IMiS®/Client is attempting to start the session with an unsupported range of encryption parameters.

WARN: "Unsupported key type (type=<value>)."
The key type IMiS®/Client is attempting to use to start a session for viewing the audit trail is not supported or allowed. The settings on the server and the client must be checked and aligned.

WARN: "Illegal length of data recieved (<decimal value>). Ignoring request <decimal value>."
When communicating with IMiS®/Client, IMiS®/ARChive Server received a request that is not of the appropriate length. The server cannot take the request and denies it. This means there is an error in IMiS®/Client. This error is only theoretical.

WARN: "Error deleting object (<decimal value>)."
An error occurred when executing a request to delete an object in IMiS®/ARChive Server. The error could be cause by incorrect changes to access rights for the file objects, or the file is not where it should be.

WARN: "Request <hexidecimal value> not expected. Ignored."
The type of request that IMiS®/ARChive Server received from IMiS®/Client was not expected for the session identified by the session handle (reference) of the client or it is irregular.
The server denied the request as invalid. These are usually requests sent by a client that is not aware that the server has been restarted.
That is why their denial by the server does not lead to improper operations.

 

WARN: “Volume “<volume description>" has no profile assigned.”
A volume which is otherwise normally entered in the internal database of IMiS®/ARChive Server has not been assigned to a profile. This is usually caused by a discrepancy in the internal server database. A specialist from the manufacturer should be consulted.

WARN: “Cannot create a socket (out of file descriptors?), error <value>: <value>.”
IMiS®/ARChive Server is using the maximum allowed number of open files. The system setting for the maximum number of open files allowed for the user with whose privileges the server is running (by default iarc ) must be increased accordingly.

WARN: “Configuration parameter ‘<parameter name>' has invalid structure and will be ignored.”
The configuration parameter in the /etc/iarc.conf file was entered incorrectly or does not match the expected range of values. IMiS®/ARChive Server ignores the settings and uses its default value.

WARN: “Error <error number> while getting object id for external id <identifier>.”
The object that was previously connected to an external identifier value <identifier> does not exist in the database.

WARN: “Error <error number> while setting external id for object <object identifier>.”
An error occurred when connecting an existing object to an external identifier. The error is only theoretical and could be the result of incorrect functioning in the internal database.
A specialist from the manufacturer should be notified.

WARN: “User <user name> from <computer name> did not authenticate with Audit Log Query permissions. Query denied!”
A user sent a request for a query of the audit log, but does not have the rights for this because the user session was not properly authenticated. This could be caused by an incorrect configuration in IMiS®/Client or it could be a warning that a particular user is attempting to perform an unauthorized action.

WARN: “semop() ended with error <error number>.”
A system error occurred when synchronizing processes. This is probably caused by incorrect functioning of the operating system or a lack of system resources. As a result, the normal operations of IMiS®/ARChive Server could be disrupted; data that have already been saved are not in danger. We recommend notifying a specialist from the manufacturer about the error and restarting the operating system (and also the service).

 

WARN: “dup(<decimal value>) error <error number>: <system description of error>.”
A system error occurred when duplicating the system identifier.
The error was probably caused by incorrect functioning of the operating system or a lack of system resources; it is described in detail in <system description of error>.
As a result, the normal operations of IMiS®/ARChive Server could be disrupted; data that have already been saved are not in danger. We recommend notifying a specialist from the manufacturer about the error and restarting the operating system (and also the service).

WARN: “Cannot put socket <decimal value> in listen mode, error <error number>: <system description of error>. Skipping to next...”
A system error occurred while listening for requests for new client connections. The error could be caused by a lack of system resources or incorrect settings in IMiS®/ARChive Server.

WARN: “Message queue full. Increase number of serving threads.”
The request queue is full. We recommend increasing the number of serving threads in the settings.

WARN: “Configured service ‘<service name>' cannot be resolved to a discreet port number (error: <system description of error>). Falling back to default ‘<service name>'...”
WARN: “Default service ‘<service name>' cannot be resolved to a discreet port number (error: <system description of error>). Default port will not be configured.”
WARN: “Default service ‘<service name>' resolves to a unsupported protocol family. Default port will not be configured.”
WARN: “Configured service ‘<service name>' resolves to a unsupported protocol family. Falling back to default ‘<service name>'...”
WARN: "Address ‘<network address>', service ‘<service name>' skipped since it cannot be resolved (error: <system description of error>).”
These warnings mean an error occurred when initializing the system for network connections. This error could be caused by an incorrect configuration in IMiS®/ARChive Server or incorrect functioning of the operating system.

WARN: “Maximum number of listening sockets reached (max = <decimal value>). Additional addresses will not be used!”
These warnings mean an error occurred when initializing the system for network connections. This error could be caused by incorrect settings in IMiS®/ARChive Server or incorrect functioning of the operating system.

WARN: “Unable to locate template <decimal value> in entity inventory. Template: Attribute bind record <decimal value> : <decimal value> will be ignored.”
The entity being searched for is not in the database.

 

WARN: #Unable to locate attribute <decimal value> in attribute inventory. Template:Attribute bind record <decimal value> : <decimal value> will be ignored."
WARN: "Configuration[ContentParsers]: Content parsing option DISABLED. Reason: '<description>'"
WARN: "Configuration[FullTextIndex]: FullTextIndex option DISABLED. Reason: '<description>'"
WARN: "Configuration[<xml tag>]: Attribute '<attribute_name>' required but missing or empty. Configuration record skipped."
WARN: "Configuration[<xml tag>]: Attribute '<attribute_name>' contains an unsupported value '<string>'. Configuration record skipped."
WARN: "Configuration[<xml tag>]: Template '<string>' not found or error accessing it. Configuration record skipped."
WARN: "Configuration[<xml_tag>]: Invalid attribute type '<decimal value>' for '<string>' (FILE required). Configuration record skipped."
WARN: "Configuration[<xml_tag>]: Configuration for template '<string>' already exists. Using first configuration record with attribute '<string>'."
WARN: "Configuration[<xml_tag>]: Unknown configuration parameter '<string>' encountered. Skipping..."
WARN: "Legacy Achival option DISABLED. Reason: '<string>'"
WARN: "Using default Security Options. Reason: '<string>'"
WARN: "Configuration[<xml_tag>]: Unknown configuration parameter '<string>' encountered. Skipping..."
WARN: "Invalid XML request: Unknown request tag '<tag name>' found, ignored."
The request contains an unexpected XML tag. This error is not critical and does not affect the operations of IMiS®/ARChive Server. The server ignores the unexpected XML tag.
This warning can be expected if the client version is newer than the server version.

WARN: "Volume info: Volume <number> not found in database!
The volume could not be found in the database. The volume settings need to be checked.

 