Functional Description

Arcsys Functional Description

25.3.1.STS
December 9, 2025

Version: 25.3.1.STS

Publication date of the document: 2025-12-09

This document is intended for anyone who requires more information on the Arcsys product.

This document is written for all owners of a valid license.

The reader agrees to respect the confidential nature of this document.

Contact details:

France	***Germany***	***USA***
INFOTEL SA *Le Valmy, 6/8/18 Avenue Léon Gaumont* *F-75020 Paris* *France*	*Insoft Infotel Software GmbH* *Sternstr. 9-11* *D-40479 Düsseldorf* *Deutschland*	*INFOTEL Corporation* *PO Box 47517 Florida* *33743 St Petersburg* *United States*
+33 (0)1 48 97 38 38	*+49 (0) 211 44 03 16-6*	*800 543 1982 – Toll-free telephone (US only)* *+1 727 343 5958*
https://techsupport.infotel.com *<software@infotel.com>*	https://techsupport.insoft-software.com *<software@insoft-infotel.com>*	https://techsupport.infotel-consulting.co.uk *<software@infotel.com>*

Table of Contents

List of Figures

List of Tables

List of Examples

Preface

1. Introduction

This document details the functional features offered by Arcsys. This document is therefore supplements the Getting Started with Arcsys guide. It offers a more specific overview of the possibilities of Arcsys.

2. Reference Documents

2.1. Concepts

Arcsys Presentation Manual: **Arcsys-presentation-25.3.1.STS-en.pdf**

Arcsys Functional Description Manual: **Arcsys-functional-description-25.3.1.STS-en.pdf**

2.2. Installing and Updating

Arcsys Prerequisites Manual: **Arcsys-requirements-25.3.1.STS-en.pdf**

Arcsys Installation Manual: **Arcsys-installation-25.3.1.STS-en.pdf**

2.3. Operations

Arcsys Administration Manual: **Arcsys-administration-25.3.1.STS-en.pdf**

Arcsys Errors Manual: **Arcsys-error-25.3.1.STS-en.pdf**

2.4. GUI

Arcsys Web Interface User Manual: **Arcsys-web-25.3.1.STS-en.pdf**

Interface Guide: **Arcsys-web-end-user-25.3.1.STS-en.pdf**

2.5. Development

Arcsys API Manual: **Arcsys-api-25.3.1.STS-en.pdf**

2.6. Option guides

ArcHP Option Guide: **Arcsys-option-archp-25.3.1.STS-en.pdf**

ArcREF Option Guide: **Arcsys-option-arcref-25.3.1.STS-en.pdf**

2.7. Optional modules

BatchReporting: **BatchReporting-UserGuide-25.3.1.STS-en.pdf**

ClassAssigner: **ClassAssigner-UserGuide-25.3.1.STS-en.pdf**

MetadataReplacement: **MetadataReplacement-UserGuide-25.3.1.STS-en.pdf**

StartRetentionDateAssigner: **StartRetentionDateAssigner-UserGuide-25.3.1.STS-en.pdf**

3. Symbols and Meanings

Note

Identifies information of particular interest

Important

Identifies important information

4. Definitions and Abbreviations

See the Glossary

Part 1. Archiving Policy

1. Definitions

1.1. Storage policy

A storage policy is a rule that is referenced by a collection. The policy dictates the storage media which are successively implemented to hold a record, as well as the retention period for each media. The storage policy is defined through the graphical interface. Applications or business users use it indirectly through the reference to a collection. A storage policy can be changed over time to reflect new retention periods or new storage media.

1.2. Retention Schedule

A retention schedule defines the start and end of record retention. It is linked to records through their class or can be specified directly at the level of each record. Modifying a retention schedule will modify the retention of the records linked to it or otherwise, depending on the user's choice. Depending on its retention schedule, a record is retained based on an event triggering the start of retention (the archive date, the start of retention or a special event) until an event triggers the end of retention (none, a time period or a specific date).

1.3. Event

An event is a fact whose date may or may not be known from which records will be retained (e.g. retention starting from the last commercial flight of the Concorde). It is used in a retention schedule as an event that triggers the start of retention.

1.4. Archiving Date / Retention Start Date

1.4.1. Definition

The **retention period** of a record is defined by its context (legal, criminal, civil, etc.) and included in Arcsys by a storage policy.

The **retention start date** is not necessarily the same as the **archiving date**; the retention start date can be before or after the archiving date.

Arcsys dissociates these two dates. The storage policy is therefore applied to the record's retention start date. The retention schedule applies either to the archiving date or the retention start date or to the event date.

1.5. Storage Pools

1.5.1. Definition

A storage pool is a storage entity addressed by the storage policy. A pool defines the duration a record spends in a given zone. The pool also defines if archiving in the addressed zone is mandatory, or the retrieval priority (case of multiple copies), etc.

The duration of the last storage pool of a series of storage pools can be limited by the retention schedule. The record will thus remain in the zone of the last storage pool until the end of its retention defined by the retention schedule.

The retention schedule takes precedence over the storage policy. A record will be retained in the zone of the last storage pool of a series of storage pools beyond the duration defined for the latter if the end of record retention has not been reached.

1.5.2. *Priority Pool*

If a record is present simultaneously in several zones and the retrieval of the record is requested, the zone related to the highest-priority pool is requested first.

If an error occurs when recovering a record in this zone, the zone related to the second-highest priority pool is then requested.

If the recovery fails in each of the zones, the retrieval of the record will fail also.

A pool has a higher priority compared to another pool if the value of its **Priority** field is smaller.

Note

- To find out the zones where a record is found, please see the Arcsys Web Interface User Manual;
- No zone can be associated more than once to a single archiving policy;
- Two zones cannot have the same retrieval priority within the same archiving policy.

1.6. Valid Archiving Policy

During archiving, the archiving policy referenced by the collection must be valid. Otherwise, archiving is not possible.

An archiving policy is said to be valid if at least one mandatory series of pools exists with a non-null duration.

Note

A series of pools is said to be mandatory if the first pool of the series whose duration is other than 0 is mandatory.

1.7. Record Migration

1.7.1. Definition

Record migration between zones is performed by an automatic Arcsys archiving engine process.

These migrations are triggered in the following cases:

- Over time (natural migration of records from one zone to another);
- When the duration in storage pools has been changed (a duration in a zone has been extended or shortened);
- On modification of the retention start date of a record.

In any case, the process analyzes the expiration date of each record to determine if migration should take place.

1.7.2. Record Expiration Date

The expiration date of each record in the current zone is recorded in the Arcsys Database and enables its migration to be triggered if appropriate.

1.8. Manifest Synchronization

1.8.1. Definition

The initial purpose of the manifest is to be able to rebuild the Arcsys Database in case of loss. It must therefore correctly mirror the database.

Moreover it is possible to edit the indexing mask at any time. Such a change inevitably leads to a modification of the metadata that are stored and found in the relational database and the manifest.

Therefore, the manifest needs to be updated after any modification of the metadata.

1.8.2. Synchronization Process

The manifest is updated through **manifest synchronization requests**. Similarly to migration requests, these requests are **automatically** created by the engine on a periodic basis, when the metadata of archived lots have been changed.

2. Strategy Defined in Arcsys

2.1. Introduction

2.1.1. Overall Strategy

The strategy implemented in Arcsys consists of considering an **overall** archiving duration (sum of the durations spent in each zone). Therefore, a change in the duration of an archiving pool will have an impact on all the records in this storage policy.

2.1.2. Details

Changing the time period of an archiving pool involves changing the expiration dates of the records in the edited pool **as well as** those of the records in the following archiving pools.

The migration process is based on the expiration dates of each record to determine if a record should be migrated (or not) from its current archiving pool (zone).

A record is never migrated to a past archiving pool.

2.2. Impact on the Following Pools

When an archiving pool's time period is changed, this automatically changes the expiration dates of the records in the edited pool as well as those of the records in **all the following** archiving pools.

2.3. Expiration Date

2.3.1. Details

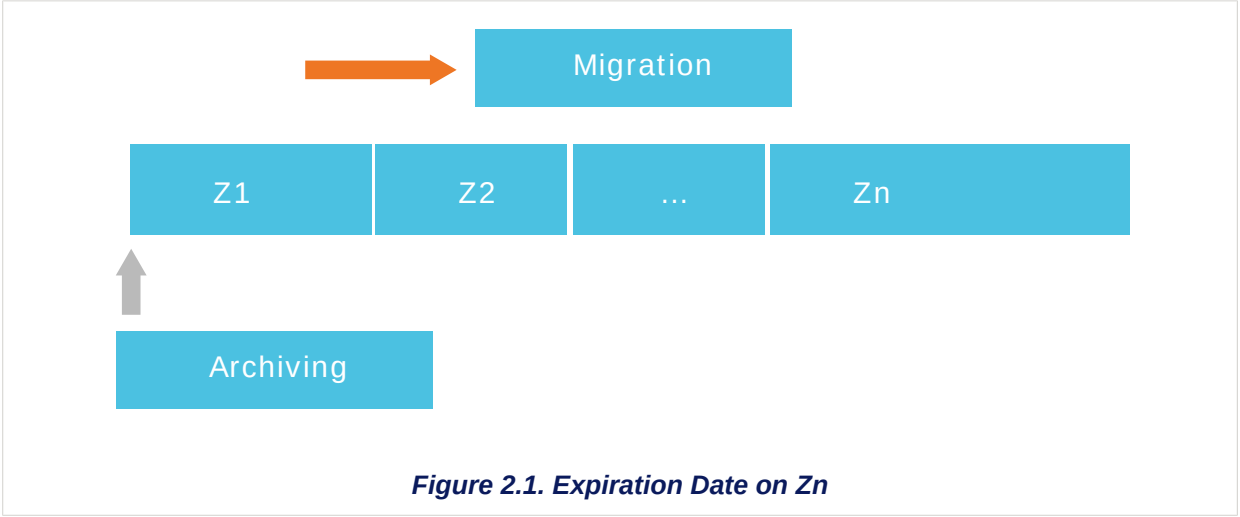

Figure 2.1. Expiration Date on Zn

When a record migrates from a source pool (here Z2) to a destination pool (here Zn), the expiration date of this record in Zn is calculated as follows:

Expiration date on Zn = retention start date + time period Z1 + … + time period Zn

2.3.2. Impact

A record that migrates towards a given pool will not necessarily remain there for the specified time period.

2.4. Placing a Disposal Hold on a Set of Records

2.4.1. Details

When a disposal hold is placed on a set of records, all operations are permitted except placing any of the held records into end-of-life status or performing migrations.

2.4.2. Impact

When a lot is released from the disposal holds, it should return to the status it would have had if the disposal hold had not been applied.

Part 2. Automatic Media Check

1. Introduction

1.1. Role of Automatic Check

1.1.1. Description

Physical storage media deteriorate due to various causes. As a result, it is imperative to regularly check the validity of the media and be able to track a log of such checks.

Arcsys automatically runs this check at configurable intervals on records chosen at random in the zones and comparing their hash value to the original one.

1.1.2. Zone Validity

Information on media validity is kept in the **zones**.

At any given time, a zone is in a given check status.

A zone in which a record could not be read correctly in the last check is thus marked with a checking error.

1.2. Zone Checking Statuses

All zone checking statuses over time can be viewed via the web interface (see Arcsys Web Interface User Manual).

A zone can have one of the following checking statuses:

- Not checked. The process has not yet checked this zone;
- OK. On its last pass, the checking process was able to correctly retrieve the lot(s) it selected, with correct integrity;
- Failed to check. A technical error has occurred, preventing the checking process;
- Error. The checking process failed to correctly retrieve a record. This means there is a media error. All records in the media are potentially threatened. More specifically, the error may be one of the following types:
  - *Media manager error* (STORAGE_DEVICE). Example: the manager does not respond,
  - *Data access error* (DATA_ACCESS). Example: the manager does not respond,
  - *Data integrity error* (DATA_INTEGRITY): the hash value of the retrieved data is not identical to the data stored in the archiving media.

2. Configuring the Checking Process

2.1. Triggering a Check

2.1.1. Parameters involved

At zone level:

Users can specify via the web interface:

- If a zone should be checked or if it should be excluded from the checking process;
- Zone checking interval (in days). This interval is understood to mean from the date when the zone was created and then from the last check date if the check was correctly passed. If the check has returned an *Error* or *Failed to check* result, the zone is systematically re-checked the next time the checking process is run.

At engine level:

The checking process starts:

- When the engine is launched;
- Then, every day at the time set in the engine properties file (**CHECKZONE_TIME_START**).

The records generated are independent of all software components. Thus the records can be reread without the need to be equipped with the software that generated them.

2.1.2. Example

The checking process is set to run every day at *20:30*.

Taking two zones:

- Z1 created on 10/08/2016 at 15:00:

  The checked frequency is 2 days:

- Z2 created on 08/08/2016 at 14:00:

  The checked frequency is 2 days.

Date	Zone Z1	Zone Z2
10/08/2016	No check.	Checked. OK.
11/08/2016	No check.	No check.
12/08/2016	Checked.Error.	Checked. OK.
13/08/2016	Checked. OK.	No check.
14/08/2016	No check.	Checked. OK.
15/08/2016	Checked. OK.	No check.

2.2. Degree of Checking: parameters involved

At engine level:

The checking process is run by selecting a set number of lots from the lots stored in the zone to be checked. The number of lots to be checked in the zone is defined by the **CHECKZONE_MAX_LOTS** parameter.

The higher this parameter is with regard to the average number of lots per zone, the greater the likelihood of detecting a problem in a record found in the zone.

Part 3. Priority of requests

1. Role of priority

The requests are loaded by Arcsys Engine from the database according to their priorities.

The possible values for the priority of a request are the integers 1 (LOW), 2 (NORMAL), 3 (HIGH) and 4 (IMMEDIATE). The higher the value, the higher the priority.

The priority IMMEDIATE does not assure that the request will be treated immediately. It depends on the priorities of the other requests and the load of the engine.

For a same given priority, the requests are sorted out by the file system identifier, the tape identifier and the mover media marker.

2. Default values of priority

The default value of the priority for a request is 2 (NORMAL).

The check, recovery, copy, synchronization, deletion and migration requests are created with the priority 1 (LOW). A deletion request may be created with another priority through Arcsys RMI API. The postpone, recycling and media migration requests are created with the priority 2 (NORMAL). The priority for the archiving, retrieval and archive restitution requests is specified in the parameters of the call for Arcsys RMI API and Arcsys REST API.

Concerning Arcsys Web Agent, the priority is set in the configuration file with the parameter *DEFAULT_PRIORITY_REQUEST* for the archiving requests and to 2 (NORMAL) for the retrieval requests.

Concerning ArcWeb Module, the priority is set to 2 (NORMAL) for archiving and retrieval requests.

Part 4. Creating and Retrieving a Record

1. Technical Conditions

1.1. Introduction

1.1.1. Description

This section describes the different technical conditions required for a successful archiving operation.

1.2. Launched Components

1.2.1. APIs

The Arcsys APIs should be launched for archiving to be successful: the status must be [STARTED_AND_WORKING].

1.2.2. Engine

The status of at least one Arcsys Engine must be [STARTED_AND_WORKING]. If this is not the case, the archiving request will not be processed.

1.2.3. Application Agent

The application agent to access data to be archived must be in [STARTED_NOT_WORKING] status.

When launching the application agent, the user must have the following rights for the data to be archived:

– Read permissions;

– Write permissions (for archiving with delete).

1.2.4. Transfer Server

The transfer server must also be launched. It will interface with the media manager to place the record on the desired physical media.

2. Functional Conditions

This section describes the different functional conditions required for a successful archiving operation.

2.1. Conditions for the Lot

2.1.1. General Description

A lot is an Arcsys entity that represents a record. In each lot, you can reference objects that correspond to the different files you wish to attach to the record.

Certain conditions are required for the lot in order for archiving to be successful.

2.1.2. The lot must be valid

For a lot to be valid, certain restrictions are required:

- It must have at least one valid object;
- The mandatory metadata inherent to the collection are informed.

Note

An object is said to be valid if the physical file it represents exists.

2.1.3. Archiving

For tracking purposes, a lot can only be archived once. If the lot is already archived (or being archived), it cannot be archived it again.

2.1.4. Retrieval or archive restitution

To retrieve a lot or to perform an archive restitution request on it, the lot must be archived.

2.2. Conditions for the Application Agent

2.2.1. Archiving

The application agent is responsible for retrieving the file data and information on the client machine and preparing the archiving. It must have access to the different files that reference the lot's valid objects.

Remember:

When launching the application agent, the user must have the following rights for the data to be archived:

– Read permissions;

– Write permissions (for archiving with delete).

2.2.2. Retrieval or archive restitution

The application agent is responsible for recreating the directory tree of the files to be retrieved based on a given directory; it also retrieves the files' rights, UID and GID, and the last access/edit date.

The agent must be launched with super user privileges to allow these modifications to take place. If this is not the case, the application agent may encounter a technical problem. However, this will not affect the retrieval (a WARN message will be generated in the logs).

2.3. Conditions for a storage policy

2.3.1. General Description

A storage policy represents an archiving policy. For the archiving to be performed correctly, the storage policy must be valid. See section Section 1.6, “Valid Archiving Policy” [3].

Part 5. ArcMover

1. Description

ArcMover is the name given to the Arcsys media manager.

It provides archiving and retrieval services to the Arcsys Transfer Server on several media types (magnetic tapes and disks).

For tape archiving, it interfaces with archiving tape libraries used to store and process magnetic tapes.

A tape library can thus be considered to be formed by magnetic tapes and a tape drive.

2. Means of Accessing the Tape library

ArcMover supports accessing the tape library with **SCSI**: "Small Computer System Interface", a set of standards for connecting and communicating between a computer and devices. The machine is directly connected to the tape library and ArcMover communicates with it via a `device Unix` file.

3. Audit Phase

3.1. Definition

The audit updates the relational database according to the information contained in the tape library.

It occurs every time the Arcsys Transfer Server is started up.

The first time the Arcsys Transfer Server is started up, it informs the database of the tape library's different components (information about changers and drives). Each time the server is started up, it will update this information according to any changes occurring since the last start-up.

3.2. Principle

The audit uses two information sources to update the Arcsys Database:

- The ArcMover configuration file;
- The information recovered from the tape library.

The Arcsys Transfer Server reads the contents of the configuration file and thus retrieves the information on the drives and the tape library to which it has access.

It retrieves the identification information for each drive and then asks the tape library for information on all its drives and on all the tapes that have already been used:

– Drives are filtered according to those present in the configuration file and integrated or updated in the relational database. The server also retrieves from the database any information on locks possibly updated by the user;

– In the database, the server updates information about all the tapes known by Arcsys (tapes that have already been used for archiving or that have been added in the database by a previous version of Arcsys Transfer Server). The Arcsys Transfer Server updates the offline status for the registered tapes.

The Arcsys Transfer Server also checks for the existence of 'Scratch Pools' whose identifier is given for each zone and for the tape library in the database. The zone/scratch pool id association is specified in the Arcsys web interface. The link between the media and the scratch pool id is made in the tape library's `<peripheral>.conf` configuration file.

Note

Even if the Arcsys Web Agent allows to assign a scratch media to a zone, this possibility should only be used by Arcsys Support Team.

3.3. File System Management

The audit procedure is different for file systems:

The Arcsys Transfer Server selects the file systems related to it (i.e. with path associated to it) in the database and checks for their existence and permissions.

As in the case of both tapes and drives, it will likewise retrieve lock information for each file system.

3.4. Important

When using ArcMover with magnetic tapes, after the first time the Arcsys Transfer Server is started up, the user must update the database via the web interface. He must then restart the Arcsys Transfer Server before launching the first archiving operation.

The user therefore **must**:

1. Check the validity of information taken from the audit;
2. Define the types of tapes associated to the drives (in the web interface);
3. Create archiving zones and specify their scratch pools (in the web interface);
4. Assign drives to the zones (in the web interface);
5. Assign tapes to the scratch pools (in the `<peripheral>.conf` configuration file).

4. Medias and Drives

4.1. Choosing the Medias

4.1.1. When Archiving

To archive a lot in a zone, the server can choose from all the media in the zone.

The choice of media is done by envelope except in the case of a spooler regroup option where the choice of media is done by chunk.

The eligible media status for this selection are *'Online'* and *'Unlocked'*.

The order of priority in selection depends on the regroup option chosen for the zone. Possible regroup options are: by `expiration date`, by `size` or by `spooler` (`expiration date` and `spooler` options are only available for ArcMover Tape Option).

Note

For a lot assigned to n zones (multiple copies), the choice is made for each zone.

Note

For ArcMover Tape Option, when a media is selected from the scratch pool, it is assigned to the zone. Since its selection, the media becomes only available for this zone. It can no longer be selected for other zones, even if they have the same scratch identifier.

4.1.1.1. Regroup by expiration date

This option is only available with ArcMover Tape Option.

This option should be selected when you want all records on a media to expire at the same period.

When the zone is set to regroup envelopes by expiration date, the order of priority when selecting a media, from highest to lowest, is:

1. non-scratch media that have enough space to store the full envelope. Among them, priority will be given to media:

   a. containing envelopes which expiration date is near the expiration date of the envelope to store

b. containing envelopes which minimal expiration date is lower than the expiration date of the envelope to store

c. containing envelopes which maximal expiration date is upper than the expiration date of the envelope to store

2. scratch media that can take the full envelope.

4.1.1.2. Regroup by size

This option allows two modes: with or without parallelization. It should be selected to minimize the number of media used (without parallelization) or to maximize the number of drives used and also fasten the archiving process (with parallelization).

When the zone is set to regroup envelopes by size *with parallelization*, the order of priority when selecting a media, from highest to lowest, is:

1. among non-scratch medias that have enough space to store the full envelope,

   a. media that is already loaded in a drive that is idle or media that is not used (neither loaded nor queued in any drive)

   b. media that is loaded in a drive and active

   c. media that is queued to be loaded in a drive, with the shortest queue

2. scratch media that can take the full envelope.

When the zone is set to regroup envelopes by size *without parallelization*, the order of priority when selecting a media, from highest to lowest, is :

1. among non-scratch medias that have enough space to store the full envelope, the one having the less available space;

2. scratch media that can take the full envelope.

4.1.1.3. Regroup by Spooler

This option is only available with ArcMover Tape Option.

This option is especially suited when working with small records (< 1Gb).

Writing small files on tape causes backhitches (hardware operation to resynchronize the position of the tape on the drive). Those backhitches take time to proceed (depending on the material), wear down the tape and impact on the size of the tape (causing blanks between blocks of data). Therefore, they should be avoided. Spooler option is intended to regroup small records in a bigger file, called chunk, and limit the number of backhitches.

When the zone is set to regroup envelopes by Spooler, the records will not be archived as soon as the request is received by the Arcsys Transfer Server. They will be cached until the actual archiving is triggered. Many options are allowed to customize the triggering of archiving:

1. `Maximal size of spool`: records will be regrouped until they reach the size specified. When this size is reached, a chunk is created and archived on the media.

2. `Spooler waiting time for a new lot`: after a record is added to the spooler, the Arcsys Transfer Server will wait for a certain delay. If no more archiving request is received on the zone at the end of this delay, the chunk will be archived whatever the value of other parameters.

3. `Maximal lot size in spooler`: allows to limit the size of lots in the spooler. If a record exceeds this size, it is archived directly on the media, as if the zone was set to regroup by size option with no parallelization.

4. `Maximal number of lots in spooler`: set the maximal number of records that will constitute the chunk.

One or more options can be ignored by setting them to `0`. When many options are set, the first value reached will trigger the archiving process of the chunk.

The archiving on the media can also be triggered at any time, whatever the value of the above parameters, by using the Arcsys Transfer Server script `flush_spool_transferServer`.

When the zone is set to regroup envelopes by Spooler, media will be selected as for regroup by size option with no parallelization.

Note

`Spooler` option impacts the writing process on the media. Since records may be cached for a while, it is important to provide the necessary space in `stage/serv` directory according to what is parametrized. See Arcsys Prerequisites Manual for more details.

`Spooler` option also impacts the reading process on the media. Accessing a record inside a chunk may take a little longer than if the record was written with another type of grouping (in addition to moving to the specified filemark, the Arcsys Transfer Server also needs to move to the specified offset within this filemark).

4.1.2. When Retrieving

Envelopes pertaining to a lot can be archived on several tapes.

In this case, the Arcsys Transfer Server will try to retrieve by selecting the media in the zone defined as top priority.

If this retrieval fails, it will then try a retrieval from the next priority zone media, and so on down to the lowest-priority zone.

4.2. Choosing the Drives

4.2.1. Selection Algorithm

The selection algorithm for drives aims to maximize reading and writing of drives and penalise auxiliary tasks (rewinding, ejection, moving between slots and drives, loading, etc.).

Consequently, it does not necessarily respect the order of arrival of archiving/retrieval requests.

4.2.2. Principle

To read/write an envelope, the server can choose from among all the drives in the envelope zone.

The drives are chosen based on the media assigned to the envelope or envelope fragment.

The eligible drives are 'Online' and 'Unlocked'.

The order of priority in selection, from highest to lowest, is as follows:

1. Drive with selected media inserted and idle;
2. Drive with selected media inserted and active (requires waiting until the end of read/write);

Note

For an archiving action: if a retrieval request on the same media is queued for this drive, the archiving request shall be placed behind the retrieval by order of processing.

For a retrieval action: if archiving on the same media is queued for this drive, the retrieval request shall be placed ahead of it in the queue by order of processing.

3. Drive without selected media inserted and idle (requires ejection then loading of new media);

4. Drive without selected media inserted and active (requires waiting until the end of read/write and ejection then loading of new media);

Note

A check request will be processed as a retrieval.

A migration request from another media manager to ArcMover will be processed as an archiving action.

A migration request from ArcMover to another media manager will be processed as a retrieval.

A migration request from ArcMover to ArcMover will be considered on two different drives (one for reading, one for writing) and considered respectively as a retrieval on the former and an archiving on the latter.

5. Information Feedback

5.1. Reporting

5.1.1. Principle

While archiving or retrieving, for each envelope, information on actions performed by the Arcsys Transfer Server is inserted in the relational database and make it possible to track the status of an archiving operation in real time from the web client.

This information will follow the processing of the task currently in progress.

It will be inserted in the following order:

1. Information about the choice of media for each of the possible envelope fragments;
2. Information about the start of writing in each fragment.

5.2. Detecting Malfunctions

5.2.1. Principle

When accessing to the equipment - tape libraries, drives, tapes - the Arcsys Transfer Server can encounter errors.

Where appropriate, it will set the item in "lock" status in the database to prevent it from being re-selected for archiving.

This lock status will be accompanied by the creation of an error in the relational database. Errors may be consulted from the web client.

Part 6. Arcsys and the Business Continuity Plan

1. Business Continuity Features offered by Arcsys

1.1. Overview

Arcsys provides features that address the following needs:

- **High availability**:
  - Arcsys can be installed on several sites so that if one site is "down", another can take its place and handle the requests originally intended for that site. The other site should have a copy of all the data present on the first site: this is **replication**.
  - Within a single site, you can install several Arcsys module instances. Thus, if one of them is unavailable, another can take over. See How many times can a module be installed on a site? for more information.
- *Load distribution*: Archiving engines can be set to deal with only certain types of requests.

1.2. Basic Concepts

1.2.1. Sites in Arcsys

1.2.1.1. Definition

A site represents the physical location where Arcsys is installed. The site can be viewed as a set of interconnected machines.

In terms of business continuity, reference is made often to the **primary site** (or "**main site**") and the **backup site** (or "**secondary site**"). Under this configuration, the backup site is used to contain the data archived in the primary site and replace it when the primary site is down. The data in the backup site and primary site are identical.

1.2.1.2. Creating Sites

The sites are entered in the Arcsys web interface and stored in the relational database.

1.2.1.3. Active or Inactive?

A site is considered **active** when the transfer server installed at the site has been started up.

Otherwise, the site is considered **inactive**.

An **inactive** site can no longer perform archiving/retrieval functions.

1.2.2. Ownership of Sites

1.2.2.1. Arcsys Modules

Some **Arcsys software components** belong to a given site. The properties file of a module is responsible for associating the modules with a given site. See Arcsys Administration Manual for more details. The components belonging to a site are:

- The Arcsys API (RMI, SOAP or REST)
- The Arcsys Transfer Server
- The Arcsys Engine
- The Arcsys Web Agent

The relational database, the application agent and the transfer server are not linked to a site.

1.2.2.2. How many times can a module be installed on a site?

Depending on the components, it is possible to install several instances of the components for a given site. However, in cases where several instances can be installed, some rules should be respected by the different component settings:

- Arcsys Transfer Server: There can be several Arcsys Transfer Servers on a given site. Linking the Arcsys Web Agent, the Arcsys Engine and the API to several Arcsys Transfer Servers, allows to address availability issues (a transfer server will process the requests if another transfer server fails).

  However, some restrictions apply using several transfer servers on a given site:

  - ArcHP Option must be used when several transfer servers share write access to a filesystem with ArcMover Disk
  - different Arcsys Transfer Servers cannot share a unique tape library managed with ArcMover Tape Option
  - the zones managed by an external media manager can be shared between several Arcsys Transfer Servers of a site only if the external media manager can handle concurrent access

- Arcsys Engine: There can be several engines on a given site. This addresses availability problems (one engine will process the requests if another one is down) as well as load distribution issues (each engine can be told what type of request to process).
- Arcsys Database: There should be only one database, possibly mirrored to the different sites.
- Arcsys API (RMI, SOAP or REST): It is possible to install several APIs on a given site. For example, you can install an API for use by specific clients.
- Arcsys Application Agent: It is possible to install several application agents on a given site. An application agent is not attached to one site.
- Arcsys Transfer Service: It is possible to install several Arcsys Transfer Services on a given site. A transfer service is not attached to a site.
- Arcsys Web Agent: It is possible to install several Arcsys Web Agents on a given site.

For more details on the limitation, see **Arcsys Administration Manual**.

1.2.2.3. Storage Zone

A **zone** belongs to a given site. A migration can only be defined between zones belonging to the same site.

However, it is possible to define a Y copy on the zones belonging to a number of different sites: this is used to **replicate data** over a number of sites, essential in implementing a recovery plan.

1.2.3. Inter-site Communication

Among the various sites configured, only communication between the **transfer servers** must be possible (preferably a high-speed link as data transfer will take place between machines hosting the transfer servers).

Application agents and transfer services that are not part of a site can communicate with engines or transfer servers, irrespective of the site of the transfer servers.

1.2.4. Architecture enabling Business Continuity

The diagram below shows the architecture used for business continuity based on:

- a primary site, e.g. involving two engines, one specializing in migration, the other for all other operations;
- a backup site.

The storage policies define Y copies to a storage zone in each of the two sites.

The relational database is natively mirrored to the two sites.

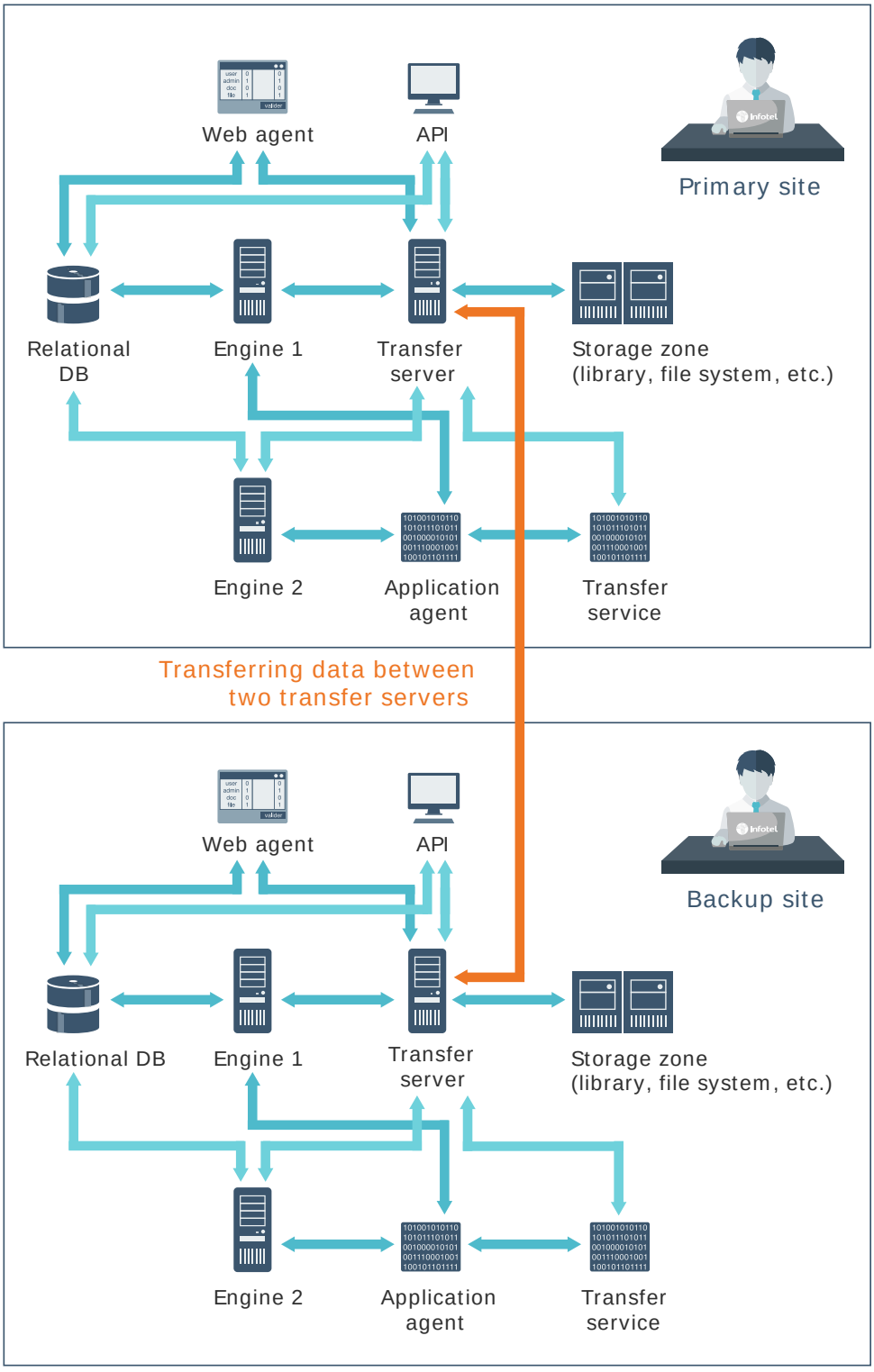

Figure 1.1. Database mirrored natively on two sites

1.3. Implementing Business Recovery

1.3.1. Definition of Recovery

Within the context of a business continuity plan, **recovery** is the process of copying data to update a site to match the data of other sites when a certain inactive period has lapsed.

1.3.2. Possibility of Recovery

An archiving request, when it involves zones belonging to several sites (in the preceding architecture example, a zone belonging to the primary site and a zone belonging to the backup site), can follow two different behaviors:

- "Successful" archiving request (allowing data retrieval) if it took place correctly on **the mandatory zones of both sites**. If one of the two sites is inactive, the request will fail and **recovery will not be possible**;
- For more tolerance toward the expected behavior: the request is successful when it is run correctly on **the mandatory zones of at least one site**. In this case, if it has failed on one of the other sites because this site is inactive, **it is possible to recover the data** from a site where archiving was correctly completed.

The choice of this behavior is dictated by the engine's **ALL_SITES_REQUIRED** parameter. See **Arcsys Administration Manual** for more details.

1.3.3. Launching a Recovery

When they start up, the engines automatically trigger the recovery process for data missing from their site: they send a data recovery order to their transfer server, which retransmits it to the transfer servers of sites where the data is correctly archived.

To launch the recovery, an engine must have the "Recovery" specialisation activated (see **Arcsys Administration Manual** for more details).

Part 7. Rights in Arcsys

1. Introduction

Arcsys offers a two-tier rights model to meet advanced partitioning needs.

Thus, in Arcsys, you must distinguish between:

- **Permissions for entities**: These permissions are linked to Arcsys entities that administrators or users create. These permissions, being associated with Arcsys entities, are stored in the *relational database*.
- **Roles**: These general rights are stored in the **LDAP directory**, with an `arcsysRight` attribute at the group level, and are independent of Arcsys entities. They grant all users who belong to these groups the right to perform certain operations (archiving, retrieval, etc.).

To determine if a user can perform an operation on the entity, check that this user has the sufficient rights on the entity and the suitable role.

1.1. Permissions for Arcsys Entities

1.1.1. General Permission Principles

Permissions apply only to Arcsys entities:

- Repositories
- Collections
- Classes
- Lots

These permissions primarily involve **visibility** or **management** rights. However, more specific functional rights may also apply, such as the ability to prepare records in a given collection or lot, or to retrieve a specific lot. These are detailed in the rights section for each entity.

Each permission can be assigned a start date and an end date. A permission is valid if the access date to the entity is on or after the start date and before the end date. If no start date is specified, the permission is valid until any specified end date. If no end date is specified, the permission is valid indefinitely from the start date. If neither date is specified, the permission is valid immediately and indefinitely.

Note

This concept of start and end dates is supported across all modules of Arcsys. However, these dates can only be configured through Arcsys REST

API or ArcWeb Module, specifically during the creation or modification of the relevant permissions.

It is also possible to define an access zone for each permission. The permission will only be valid if, during authentication to Arcsys REST API, Arcsys Web Agent, or ArcWeb Module, the client's IP address is validated against the access zone associated with the permission. For more details, see page 122, « Access zones ».

1.1.2. Default Permissions and Archive Specifics

A user who creates an entity automatically acquires management permissions for this entity. Users cannot delete their own permissions.

For archives, this behavior is governed by a collection-level setting that determines whether the permissions assigned to a lot before archiving are retained. By default, these permissions are preserved.

If the collection is configured to discard pre-archiving permissions, they will be removed at the end of the archiving process. In that case, the permissions defined by the applicable lot permission templates are applied instead.

1.2. Roles

There are several role types in Arcsys:

- The **GRANT_ALL** role grants a **super-user** right to the group that owns such right: it can impact all operations on all entities.

Important

From the moment a user performs "GRANT_ALL" on an operation, no rights checks are performed.

- The **ADMIN** role is used to modify the rights of a repository, a collection or a class without having visibility of it. But as opposed to GRANT ALL, **it does not give visibility of records**. This right is also used to view and manage technical Arcsys components (engines, application agents, transfer servers, transfer services). Moreover, only an ADMIN (or GRANT_ALL) user can see technical domain entities.

- "Management" roles: **BASE_MANAGEMENT, COLLECTION_MANAGEMENT, INDEXATION_MANAGEMENT, STORAGE_MANAGEMENT, LIFECYCLE_MANAGEMENT, WORKFLOW_MANAGEMENT, ENCRYPTION_MANAGEMENT, ACCESS_ZONE_MANAGEMENT**. These roles are used, for example, to manage administration entities (repository, collection, keywords, masks, storage, retention schedules and workflows). They are not necessary for a user who must perform archive or retrieve operations on a daily

basis. Having disassociated rights means you can establish different roles. Distribution examples are shown below.

- The **PREPARE_ARCHIVE_OBJ, ARCHIVE, RESTORE, DISPOSAL_HOLD** roles, relating to record preparation, record archiving and retrieval operations, record disposal.
- The **PROOF_FOLDER** role, permitting to generate proof folder or proof slip.
- The **WORKFLOW** role, permitting to approve or reject a workflow.
- The **OBJ_DELETE** role, permitting to delete an object from an archived lot.
- The **LOT_ENRICHMENT** role, permitting to manage lot enrichment operations.
- The **FUNCTIONAL_ADMIN** role, giving access to the functional administration page in the home menu of ArcWeb (Arcsys entities management).
- The **ARCWEB_ADMIN** role, giving access to the ArcWeb settings pages in the user-level menu (theme settings, group customization, etc.).

In the Arcsys Web Agent, if the user has the ADMIN, GRANT_ALL role or a management role, the user is directed to the Administration Web interface. Otherwise, the user is directed to the End User Web interface.

2. Table of Rights Required by Entities

For this purpose, the term "visibility" for an entity refers to viewing (in API or web display mode) the characteristics of an entity. In the absence of suitable permissions, users cannot see this entity, whether they ask for this explicitly via identifier or whether they get a list of entities.

The term "management" of an entity refers to a creation, modification or deletion of this entity.

2.1. Functional Administration

2.1.1. Repositories, Collections and Classes

Roles	Description
GENERAL ROLE	
ADMIN	Used to view a repository, a collection or a class, or modify its rights without the visibility right.
REPOSITORIES	
BASE_MANAGEMENT	To manage a repository (previously called 'base').
COLLECTIONS	
COLLECTION_MANAGEMENT	To manage a collection.
CLASSES	
There are no specific roles required for class management.	

Table 2.1. Roles

Permission	Description
REPOSITORIES	
Management Abbreviation: Repository: M	Can change the characteristics of the repository or delete it. Giving Management permission automatically gives Visibility permission on the repository.
Visibility Abbreviation: Repository: V	View the repository and use it to create other entities.
COLLECTIONS	
Management Abbreviation: Collection: M	Can change the characteristics of the collection or delete it. Giving Management permission automatically gives Visibility permission on the collection.
Visibility Abbreviation: Collection: V	View the collection and use it to create other entities.

Permission	Description
Prepare archive Abbreviation: Collection: P	Create lots and objects in this collection. Giving Prepare archive permission automatically gives Visibility permission on the collection.
CLASSES	
Management Abbreviation: Class: M	Can change the characteristics of the class or delete it. Giving Management permission automatically gives Visibility permission on the class.
Visibility Abbreviation: Class: V	View the class and therefore any records that are associated with it.
Edition Abbreviation: Class: E	Add or remove the child class or delete the parent class. Giving Edition permission automatically gives Visibility permission on the class.

Table 2.2. Permissions

General operation (API or web)	Roles and permissions required
REPOSITORIES	
View a repository	ADMIN role or (Repository: V)
GUI: Access to the base administration screen	ADMIN or BASE_MANAGEMENT role
Create a repository	BASE_MANAGEMENT role
Modify, delete a repository	Creation rights + (Repository: V, M)
Operations on repository permissions	ADMIN role or (Modification rights)
COLLECTIONS	
View a collection	ADMIN role or (Repository/Collection: V)
GUI: Access to the collection administration screen	ADMIN or COLLECTION_MANAGEMENT role + (Repository: V)
Create a collection	COLLECTION_MANAGEMENT+ role (Repository: V)
Modify, delete a collection	Creation rights + (Collection: V,M)
Operations on collection permissions	ADMIN role or (Modification rights)
CLASSES	
View a class	ADMIN role or (Repository/Class: V)
GUI: Access to the administration screen for classification schemes	ADMIN or BASE_MANAGEMENT role
Create a class	Repository: M
Modify the hierarchy of classes (children, parent)	Repository: V, Class: E
Modify, delete a class	Repository: V, Class: M
Operations on class permissions	ADMIN role or (Modification rights)

Table 2.3. Summary: Rights Depending on Roles and Permissions

2.1.2. Indexing

The general term "indexing entity" refers to one of the following entities: masks, keywords, keywords in a mask, values in a keyword.

Role	Description
INDEXATION_MANAGEMENT	To manage indexing

Table 2.4. Role

Permission	Description
INDEXING ENTITIES	
There are no specific permissions associated with indexing entities (they are provided by the repository to which they are attached).	

Table 2.5. Permissions

General operation (API or web)	Roles and permissions required
View an indexing entity	Repository: V
GUI: Access to all the screens relating to indexing (lists of masks, keywords, keywords in a mask, values in a keyword, etc.)	ADMIN role or (Repository: V)
Manage an indexing entity	INDEXATION_MANAGEMENT role + Repository: V

Table 2.6. Summary: Rights Depending on Roles and Permissions

2.1.3. Retention

The general term "retention entity" refers to one of the following entities: retention schedule, event.

Role	Description
LIFECYCLE_MANAGEMENT	To manage retention (retention schedule, event).

Table 2.7. Roles

Permission	Description
RETENTION ENTITIES	
There are no specific permissions associated with retention entities (they are provided by the repository to which they are attached).	

Table 2.8. Permissions

General operation (API or web)	Roles and permissions required
View a retention entity	Repository: V

General operation (API or web)	Roles and permissions required
GUI: Access to the administration screen for retention entities	LIFECYCLE_MANAGEMENT role
Manage a retention entity	ADMIN role or (LIFECYCLE_MANAGEMENT role + Repository: V)

Table 2.9. Summary: Rights Depending on Roles and Permissions

2.1.4. Format policy

Role	Description
ADMIN	To manage format policies.

Table 2.10. Roles

Permission	Description
FORMAT POLICIES	
There are no specific permissions associated with format policies (they are provided by the repository to which they are attached).	

Table 2.11. Permissions

General operation (API or web)	Roles and permissions required
View a format policy	Repository: V
Manage a format policy	ADMIN role

Table 2.12. Summary: Rights Depending on Roles and Permissions

2.1.5. Attestation policy

Role	Description
ADMIN	To manage attestation policies.

Table 2.13. Roles

Permission	Description
ATTESTATION POLICIES	
There are no specific permissions associated with attestation policies (they are provided by the repository to which they are attached).	

Table 2.14. Permissions

General operation (API or web)	Roles and permissions required
View an attestation policy	Repository: V

General operation (API or web)	Roles and permissions required
Manage an attestation policy	ADMIN role

Table 2.15. Summary: Rights Depending on Roles and Permissions

2.1.6. Workflow policy

Role	Description
WORKFLOW_MANAGEMENT	To manage workflow policies.

Table 2.16. Roles

Permission	Description
WORKFLOW POLICIES	
The Repository: M permission is required to administrate workflow policies.	

Table 2.17. Permissions

General operation (API or web)	Roles and permissions required
View a workflow policy	Repository: V
Manage a workflow policy	WORKFLOW_MANAGEMENT role + Repository: M

Table 2.18. Summary: Rights Depending on Roles and Permissions

2.1.7. Encryption management

Role	Description
ENCRYPTION_MANAGEMENT	To view or manage encryption policies, to view or manage the encryption key of a collection, to view or manage the key managers and the master keys, for master key rotation.

Table 2.19. Roles

Permission	Description
ENCRYPTION POLICIES	
There are no specific permissions associated with encryption policies (they are provided by the repository to which they are attached).	
ENCRYPTION KEYS	
There are no specific permissions associated with encryption keys (they are provided by the collection to which they are attached).	
KEY MANAGERS AND MASTER KEYS	
There are no specific permissions associated with key managers and master keys.	

Table 2.20. Permissions

General operation (API or web)	Roles and permissions required
View or manage an encryption policy	ENCRYPTION_MANAGEMENT role
View or manage the encryption key for a collection	ENCRYPTION_MANAGEMENT role + Collection: V
View or manage the key managers and master keys	ENCRYPTION_MANAGEMENT role
Trigger a master key rotation	ENCRYPTION_MANAGEMENT role

Table 2.21. Summary: Rights Depending on Roles and Permissions

2.1.8. Proof folder

Role	Description
PROOF_FOLDER	To be able to generate proof folders or proof slips.

Table 2.22. Roles

Permission	Description
PROOF FOLDER GENERATION	
There are no specific permissions associated with proof folder generation.	

Table 2.23. Permissions

General operation (API or web)	Roles and permissions required
Generate a proof folder (or proof slip)	PROOF_FOLDER role + Repository/Collection/Lot: V

Table 2.24. Summary: Rights Depending on Roles and Permissions

2.1.9. Access zone management

Role	Description
ACCESS_ZONE_MANAGEMENT	To manage access zones.

Table 2.25. Roles

Permission	Description
ACCESS ZONES	
There are no specific permissions associated with access zones.	

Table 2.26. Permissions

General operation (REST API)	Roles and permissions required
Manage an access zone	ACCESS_ZONE_MANAGEMENT role

Table 2.27. Summary: Rights Depending on Roles and Permissions

2.2. Technical Administration

2.2.1. Storage

Role	Description
STORAGE_MANAGEMENT	To manage the storage policy.

Table 2.28. Roles

Permission	Description
STORAGE POLICIES	
There are no specific permissions associated with storage policies (they are provided by the repository to which they are attached).	
ZONES	
There are no specific permissions associated with storage zones.	
SITES	
There are no specific permissions associated with storage sites.	
STORAGE MEDIA (ARCMOVER)	
There are no specific permissions associated with storage media.	

Table 2.29. Permissions

General operation (API or web)	Roles and permissions required
STORAGE POLICIES	
View a storage policy	Repository: V
GUI: Access to the administration screen for storage policies	ADMIN role or Repository: V
Manage a storage policy	STORAGE_MANAGEMENT role + Repository: V
ZONES	
View and manage storage zones	STORAGE_MANAGEMENT role
GUI: Access to the administration screen for storage zones	STORAGE_MANAGEMENT role
SITES	
Manage a site	STORAGE_MANAGEMENT role
GUI: Access to the administration screen for sites	STORAGE_MANAGEMENT role
MEDIA (TAPE DRIVES, TAPE LIBRARIES, FILESYSTEMS, etc.)	
Manage media	STORAGE_MANAGEMENT role

Table 2.30. Summary: Rights Depending on Roles and Permissions

2.2.2. Technical Operating Activities

General operation (API or web)	Roles and permissions required
GUI: Access to the list of Arcsys modules (Engines, Application Agents, Transfer Servers, Transfer Services)	ADMIN role
GUI: Access to the list of lots pending recovery	ADMIN role
GUI: Access to the list of requests	ADMIN role
GUI: Access to volume screens	ADMIN role

2.3. End user

2.3.1. Lots, Objects, Metadata and Association with Classes

Role	Description
GENERAL ROLE	
ADMIN	Used to modify the permissions of a lot without the viewing permission.
LOTS	
PREPARE_ARCHIVE_OBJ	To manage a lot, an object, a metadata or a link between a class and a lot.
OBJ_DELETE	To delete objects from archived lots.
LOT_ENRICHMENT	To manage lot enrichment operations.

Table 2.31. Roles

Permission	Description
LOTS	
Management Abbreviation: Lot: M	To change the characteristics of the lot or delete it. Giving Management permission automatically gives Visibility permission on the lot.
Visibility Abbreviation: Lot: V	View the lot and use it to create other entities.
Archive Abbreviation: Lot: A	Create an archiving request on a lot. Giving Archive permission automatically gives Visibility permission on the lot.
Retrieval Abbreviation: Lot: R	Create an asynchronous retrieval request on a lot, or create an archive restitution request on a lot. Giving Retrieval permission automatically gives Visibility permission on the lot.
Download Abbreviation: Lot: D	Create a synchronous retrieval request on a lot. Giving Download permission automatically gives Visibility permission on the lot.

Permission	Description
OBJECTS	
There are no specific permissions associated with objects (they are provided by the lots to which they are attached).	
METADATA	
There are no specific permissions associated with metadata (they are provided by the lots to which they are attached).	

Table 2.32. Permissions

General operation (API or web)	Roles and permissions required
LOTS	
View a lot	Repository/Collection/Lot: V.
Create a lot	PREPARE_ARCHIVE_OBJ role + Repository/Collection: V, Collection: P
Edit a lot	Creation rights + Lot: V, M
Delete a lot	PREPARE_ARCHIVE_OBJ role + Repository/Collection/Lot: V, Lot: M
Archive or retrieve a lot	See following paragraph "Archiving or Retrieval Request"
OBJECTS	
View an object (or its format characteristics)	Repository/Collection/Lot: V
Create, Edit an object on a lot that is not archived	PREPARE_ARCHIVE_OBJ role + Repository/Collection: V, Lot: M
Create, Edit an object on a lot that is archived	LOT_ENRICHMENT role + Repository/Collection: V, Lot: M
Delete an object from a lot that is not archived	PREPARE_ARCHIVE_OBJ role + Repository/Collection: V, Lot: M
Delete an object from a lot that is archived	OBJ_DELETE role + Repository/Collection: V, Lot: M
METADATA	
View metadata	Repository/Collection/Lot: V
Manage metadata	PREPARE_ARCHIVE_OBJ role + Repository/Collection: V, Lot: M
LOT CLASS ASSOCIATIONS	
View the link between a class and a lot (if you prevent viewing between a class and a lot, you do not necessarily prevent viewing of the lot if you have visibility for this lot.)	Repository/Collection/Lot/Class: V
Manage the link between a class and a lot	PREPARE_ARCHIVE_OBJ role + Repository/Collection: V, Lot: M

Table 2.33. Summary: Rights Depending on Roles and Permissions

2.3.2. Archiving or Retrieval Request

Role	Description
ARCHIVE	To archive a lot
RESTORE	To retrieve a lot or an object, or to create an archive restitution request on a lot.
DISPOSAL_HOLD	To hold a lot
LOT_ENRICHMENT	To create an enrichment request on an archived lot

Table 2.34. Roles

Permission	Description
REQUESTS	
There are no specific permissions associated with requests (they are provided by the lots to which they are attached).	
DISPOSAL HOLDS	
There are no specific permissions associated with record holds (they are provided by the lots to which they are attached).	

Table 2.35. Permissions

General operation (API or web)	Roles and permissions required
REQUESTS	
View a request	Repository/Collection/Lot: V.
Create an archiving request	ARCHIVE role + Repository/Collection/Lot: V + Lot : A
Create an asynchronous retrieval request or an archive restitution request	RESTORE role + Repository/Collection/Lot: V + Lot : R
Create a synchronous retrieval request	RESTORE role + Repository/Collection/Lot: V + Lot : D
Create an enrichment request	LOT_ENRICHMENT role + Repository/Collection/Lot: V + Lot : A
Manage a request	ARCHIVE role for archiving (or RESTORE for retrieval or archive restitution) + Repository/Collection/Lot: V, Lot: M
DISPOSAL HOLDS	
View a disposal hold	Repository/Collection/Lot: V
Manage a hold	DISPOSAL_HOLD role + Repository/Collection/Lot: V, Lot: M

Table 2.36. Summary: Rights Depending on Roles and Permissions

2.3.3. Approve or reject a workflow

Role	Description
WORKFLOW or WORKFLOW_MANAGEMENT	To list the lots concerned by a workflow: workflow pending, approved or rejected.
WORKFLOW	To approve a workflow.
WORKFLOW and DISPOSAL_HOLD	To reject a workflow.

Table 2.37. Roles

Permission	Description
LOTS	
For a user to list the lots in workflow pending status, the Visibility permission on the lot is required.	

Table 2.38. Permissions

General operation (API or web)	Roles and permissions required
View the lot in workflow pending status	(WORKFLOW or WORKFLOW_MANAGEMENT) + Lot: V.
Approve or reject a workflow	WORKFLOW role + Lot: V,M + be member of the group defined in the task scheme of the workflow policy

Table 2.39. Summary: Rights Depending on Roles and Permissions

3. Overview: Group Implementation Examples

This section provides a number of implementation examples for allocating rights at LDAP directory level, using simple use cases.

3.1. Single Super User

In a context where there is no point in distinguishing between users, a single group suffices (e.g. ARCSYS_USERS), which has the GRANT_ALL role.

3.2. Administrator and Users

If a basic division between groups is desired, you can define two groups:

- A group of administrators,e.g. ADMIN_USERS, which has the GRANT_ALL role;
- A group of end users, e.g. END_USERS, which has the following roles: PREPARE_ARCHIVE_OBJ, ARCHIVE, RESTORE, DISPOSAL_HOLD.

3.3. Technical Administrator, Functional Administrators and Users

If you want to distinguish between two administrator classes, i.e. administrators in charge of the technical domain (components, storage), and administrators in charge of the functional domain (repositories, collections, keywords, retention schedule; record manager role), you can create the following:

- A group of technical administrators, e.g. ADMIN_TECHNICAL_USERS, which has the following roles: ADMIN, STORAGE_MANAGEMENT;
- A group of functional administrators,e.g. ADMIN_FUNCTIONAL_USERS, which has the following roles: BASE_MANAGEMENT, COLLECTION_MANAGEMENT, LIFECYCLE_MANAGEMENT, INDEXATION_MANAGEMENT, WORKFLOW_MANAGEMENT, PREPARE_ARCHIVE_OBJ, ARCHIVE, RESTORE, DISPOSAL_HOLD, PROOF_FOLDER, WORKFLOW, OBJ_DELETE, LOT_ENRICHMENT;
- A group of end users, e.g. END_USERS, which has the following roles: PREPARE_ARCHIVE_OBJ, ARCHIVE, RESTORE, DISPOSAL_HOLD.

Part 8. End User Web Interface

1. About Rights

1.1. Introduction

Depending on the group to which they belong, users will be granted different rights.

In the web interface, these rights define the interface to which users are sent after connection (classic ‘administrator’ interface or end user interface) and the functions they are granted.

The rights a user can have are detailed in the section ‘Managing Rights’.

1.2. Redirect

If a user has **at least one of the following rights**, they will be considered an ‘**administrator**’ and will then be **redirected to the Administrator interface** after connection:

– **BASE_MANAGEMENT**

– **COLLECTION_MANAGEMENT**

– **STORAGE_MANAGEMENT**

– **INDEXATION_MANAGEMENT**

– **ADMIN**

– **GRANT_ALL**

If not, they will be considered an ‘**end user**’ and will then be **redirected to the End User interface** after connection.

2. Configuration

2.1. Introduction

The **End User Web Interface** requires a specific network configuration.

Note

In this chapter, **link** means a symbolic link (Unix environment) or a network drive (Windows environment) that **links a site on a machine to an access name.**

2.2. Required Configuration

The network configuration must follow these rules:

- Operating system rules:

  In **Windows**, a user can archive data from a **Windows** (network drive) or **Unix** (Samba) machine, but the data will only be processed by a **Windows** agent.

  In **Unix**, a user can archive data from a **Windows** (Samba) or **Unix** (Unix assembly) machine, but the data will only be processed by a **Unix** agent.

- Rules on **links**:

  An agent can archive data from **several different machines** by simply *specifying* the **links** to these machines.

  The *links* specified to the agents must be **the same** as those specified to the client machines.

  A *link* should be **unique** and must always reference a location on a machine. This location should also be **unique**: a *link* cannot point to two different machines or two different directories.

  Several agents may be found on the same machine and thus "linked" to the same data machines.

  A single data machine may be known by several different agents.

2.3. Examples

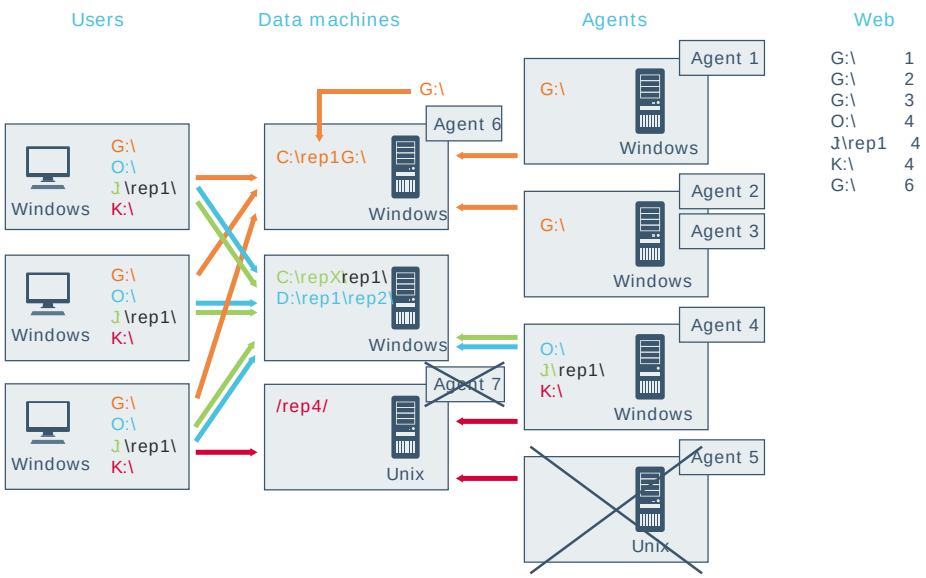

Figure 2.1. Example of configuration with users under Windows

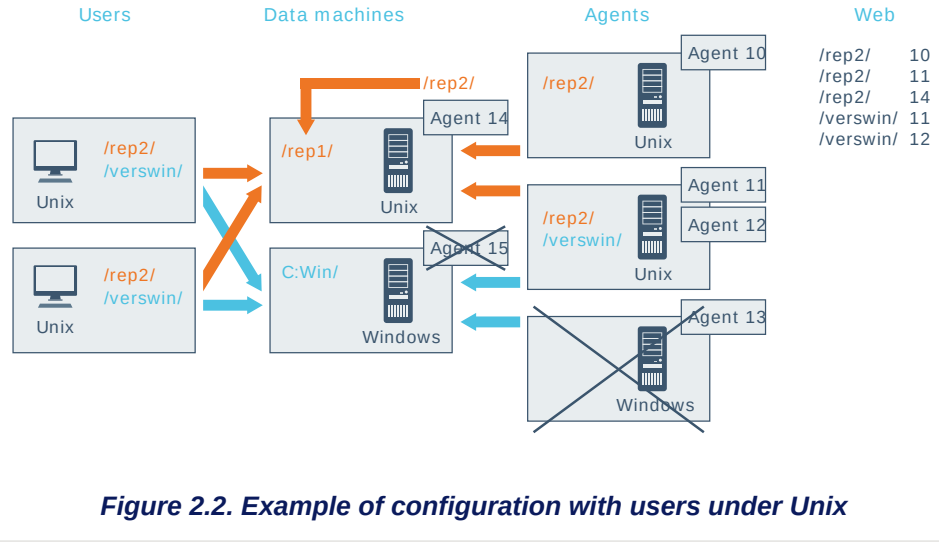

Figure 2.2. Example of configuration with users under Unix

2.4. Impact

2.4.1. Creating a Lot

- When creating a lot, the link is automatically identified by the first element added.

  Example:

- Adding of: *C:\dir\file1*
- Known link: *C:\dir*
- Link of new lot: *C:\dir*

- All objects in the same lot must be associated with the same *link*

2.4.2. Archiving

- The lot's *link* is identified by the objects contained in it.
- The archiving agent is chosen to regulate the loads between the possible agents for the lot's *link*.

2.4.3. Retrieval or archive restitution

- The retrieval agent is chosen to regulate the loads between the possible agents for the lot's *link*.
- The destination must necessarily begin as defined in the web properties file for the chosen agent and *link*.

Part 9. Classification Scheme

1. Classification Scheme

1.1. Overview

1.1.1. Introduction

Arcsys offers the ability to manage a classification scheme in the form of a tree structure.

This feature conforms to the chapter on the hierarchical classification scheme in the MoReq2010 specification.

1.1.2. Classes

A hierarchical classification scheme is a class tree.

A class can be a child class or a root class, depending on whether or not it has a parent class.

There are no limitations as to the size of the tree. This means that:

- A classification scheme can have any number of root classes;
- A class can have any number of child classes;
- The tree depth is not limited.

1.1.2.1. Path of a Class

Each class has a name ("subtitle" under MoReq2010) which determines the full path ("title" under MoReq2010) of the class as well as its child classes.

The full path of a class is the concatenation of the full path of the parent class with the class name.

Example:

If a class is named *Access and Equity* and is the child of a class whose full path is *Community Relations/Meetings*, the full path would be *Community Relations/Meetings/Access and Equity*.

1.1.2.2. Notes

The notes of a class ("textual scope notes" under MoReq2010) are intended to explain to users how the class should be used for classification.

1.1.2.3. Status

A class is active on creation, but it can then be disabled. A disabled class cannot have active child classes; disabling involves an entire branch of the classification scheme tree.

1.2. Operational Context

1.2.1. Delimiters

The repository in the Arcsys Database is the highest level of logical structure. Each repository has a set of data that belongs to it (keywords, storage policies, collections, etc.). The classification scheme is included in this set of data.

This means that:

- A class associated with a repository can only have classes associated with the same repository as child classes;
- There can be as many classification schemes as there are repositories in a database (the classification scheme is optional).

2. Classification

2.1. Association

2.1.1. Overview

The purpose of a classification scheme is to propose a logical organization of records, contrary to collections which organize records depending on indexing and storage policy.

A lot can only be associated with a single class.

2.1.2. Limitations

It is not possible to associate a lot or an object with any class. A class is only eligible if:

- It belongs to the same repository as the lot;
- It has no child classes;
- It is active.

Classification is an important operation in Arcsys. When a class has been used to classify a lot, it cannot be deleted. However, it can be disabled in order to prevent future classifications.

2.2. Representation

Once the lots/objects have been classified, they become the representation of the classification scheme.

The classification scheme is thus depicted as a tree whose nodes are the classes and whose leaf nodes are the lots.

This depiction provides a different way of browsing through the records in a repository.

Part 10. Functional Traces

1. Generating Traces

Arcsys allows functional traces to be generated and categorized by repository and/or collection.

A trace file is generated in a specific directory by the `trace/` module.

Generating traces or not for a given repository or collection can be configured independently for any of the these entities. Traces may be generated either from Arcsys Engine, Arcsys RMI, TCP/IP and SOAP API, or Arcsys Web Agent.

Important

When traces must be generated while performing an operation, an error is raised if an impossibility to write the trace is encountered (for example due to insufficient system permissions on the trace file).

1.1. When are traces generated ?

Repository traces are generated in the following cases:

- When creating, modifying or deleting a repository
- When creating or modifying an attestation policy
- When attaching an attestation policy to a class
- When detaching an attestation policy from a class
- When creating or modifying a format policy
- When attaching a format policy to a class
- When detaching a format policy from a class
- When creating, modifying or deleting a retention schedule
- When attaching a retention schedule to a class
- When detaching a retention schedule from a class
- When creating or modifying an event

Collection traces are generated in the following cases:

- When creating, modifying, deleting a collection (thus deleting a repository as it deletes its collection in cascade) or its permissions

- When adding, updating or removing a collection encryption key
- When creating, modifying, deleting a lot or its permissions
- When creating, modifying, deleting a retention hold on a lot
- When modifying indexing information, such as:
  - When modifying or deleting a mask, when attaching or detaching a keyword from a mask
  - When modifying or deleting a keyword
  - When attaching or detaching a controlled value from a keyword
  - When creating, modifying or deleting metadata
- When modifying profile information, such as:
  - When modifying or deleting a profile, when attaching or detaching a zone from a profile
  - When modifying a pool (characteristics of a zone in a profile)
- When creating, modifying, deleting an object or the format information about an object
- When creating, modifying, deleting a request

1.2. Trace Details

Traces store the following information for creation/editing/deletion purposes:

- all information about the repository
- all information about the collection
- all information about the lot
- all information about the objects
- all information about the metadata
- all information about the keyword
- all information about the mask/keyword association
- all information about the keyword/value association

- all information about the attestation policy
- all information about the format policy
- all information about the retention schedule
- all information about the event
- all information about the class/attestation policy association
- all information about the class/format policy association
- all information about the class/retention schedule association
- for an archiving request:
  - request type
  - lot:
    - lot identifier
    - lot code
    - lot size
    - metadata
  - objects:
    - object type
    - object code
    - hash value
    - hash function
    - object size
    - metadata
  - user
  - creation time
  - for the terminal status, time and status
- for a retrieval or archive restitution request:
  - request type

- lot:
  - lot identifier
  - lot code
- the objects, in case of partial or synchronous retrieval:
  - object type
  - object code
- user
- creation time
- for the terminal status, time and status

2. Archiving Traces

Arcsys allows the generated traces to be archived with the Arcsys Auto-Archive Agent.

2.1. Archiving collection traces

You can archive collection traces by launching the trace auto archiving script that is found in the Arcsys Auto-Archive Agent.

2.1.1. Architecture

For each collection with generated traces, the module creates a lot (called a "trace lot") in the collection, in order to follow the storage policy of the collection .

This lot is made up of generated trace files and is automatically archived.

2.2. Archiving repository traces

Contrary to the traces by collection, to archive repository traces, it is necessary to configure and to use the log auto archiving script that can be found in the Arcsys Auto-Archive Agent.

Note

These traces should be archived in a collection of the `AdminBase` repository dedicated to them, with a permanent retention or longer than the longest archive retention present in Arcsys.

2.2.1. Configuration

To configure the Arcsys Auto-Archive Agent to archive repository traces, a new `AutoArchive.properties` file dedicated to the archiving of these traces should be created as well as a new `launch_autoArchive` script referencing this file via the *`arcsys.config`* VM parameter. In this new configuration file, the *`REFERENTIAL_USED_COLLECTION`* parameter should specify the appropriate collection.

Then there are several ways to fill in the other parameters to archive the repository traces:

- The first solution is to specify the *`X_ARCHIVE_FILES`* parameter with a file path pattern like **../<component_folder>/trace/repository_*.log.***. Below is an example of archiving parameters to archive traces of Arcsys Engine, Arcsys REST API and Arcsys Web Agent from all repositories:

```
1_ARCHIVE_FILES=../ArcsysEngine/trace/repository_*.log.*
1_ARCHIVE_LASTMODIFIED=1
1_ARCHIVE_SERVICE_NAME=EngineService
1_ARCHIVE_COMPONENT_TYPE=Engine

2_ARCHIVE_FILES=../ArcsysApiRestWS/trace/repository_*.log.*
2_ARCHIVE_LASTMODIFIED=1
2_ARCHIVE_SERVICE_NAME=ApiRestWSService
2_ARCHIVE_COMPONENT_TYPE=ApiRestWS

2_ARCHIVE_FILES=../ArcsysWebAgent/trace/repository_*.log.*
2_ARCHIVE_LASTMODIFIED=1
2_ARCHIVE_SERVICE_NAME=WebAgentService
2_ARCHIVE_COMPONENT_TYPE=WebAgent
```

- Another solution is to create a block of parameters dedicated to archiving (*X_ARCHIVE_FILES*, *X_ARCHIVE_LASTMODIFIED*, *X_ARCHIVE_SERVICE_NAME*, *X_ARCHIVE_COMPONENT_TYPE*) for each repository by specifying the identifier of the repository in the file path pattern specified in the *X_ARCHIVE_FILES* parameter. Then it is possible, for each repository block, to specify the corresponding repository code in the *X_ARCHIVE_SERVICE_NAME* parameters. The advantage of such a solution is that it is then possible to search for all traces of a specific repository by filtering on the `ServiceName` metadata.

  Below is an example of archiving parameter block to archive traces of Arcsys Engine from repository with identifier **3** and code **ThirdRepository**:

```
1_ARCHIVE_FILES=../ArcsysEngine/trace/repository_3.log.*
1_ARCHIVE_LASTMODIFIED=1
1_ARCHIVE_SERVICE_NAME=ThirdRepository
1_ARCHIVE_COMPONENT_TYPE=Engine
```

Part 11. Journals

1. Introduction

Arcsys is capable of generating journals by repository. These journals are XML files that contain PREMIS v3 events.

For more information on PREMIS, please visit https://www.loc.gov/standards/premis/v3/index.html.

The PREMIS v3 events relative to a repository are inserted in the database provided that event recording is enabled at repository level, either through the web interface, or via the API.

Important

For security purposes, once event recording is activated for a specific repository, it cannot be disabled.

2. Components and roles regarding journals

The following components are capable of **recording events** in Arcsys Database:

- Arcsys API (RMI, SOAP or REST)
- Arcsys Engine
- Arcsys Web Agent

The **generation of the journal** from the events recorded in Arcsys Database must be done through scripts available in Arcsys Engine.

The archival of the journal and **verification of the conformity of the chaining process** are conducted by an external process not included in Arcsys. Throughout the rest of this document, we will refer to this process as `journalArchiver`.

3. Chaining process

In summary, the chaining process involves including the fixity check of the preceding journal as the initial event in each subsequent journal.

The steps for managing journal chaining are outlined as follows:

1. Events are continuously generated and stored in the database.
2. At a certain point, a journal file generation process is initiated by executing the `journal.sh(.bat)` script from Arcsys Engine.
3. This process retrieves events from the database, arranges them in chronological order, and generates a new journal file. Additionally, a fixity check (checksum) is computed for this file.
4. The checksum of the newly created journal file (referred to as`journal_repo1_1.xml` in this example) is stored in the database as an event.
5. The resulting journal file `journal_repo1_1.xml` is archived using `journalArchiver`.
6. When the subsequent journal (`journal_repo1_2.xml`) is generated, its first event is the generation of the previous journal. This event includes the checksum of the file.
7. `journalArchiver` employs the API to access the checksum of the `journal_repo1_1.xml` journal, parses the `journal_repo1_2.xml`, searches for the “message digest calculation” event type, and confirms its reference to `journal_repo1_1.xml`.
8. Subsequently, `journalArchiver` compares the checksums. If they match, the processing and archival of the journal continue; if not, the chain is deemed broken, and `journalArchiver` halts.

4. List of events in journals

4.1. Events

The following table lists the events that are generated in journals:

- The first column represents the event name;
- The second column indicates the corresponding PREMIS v3 eventType in the generated XML file;
- The third column specifies the associated objects (according to PREMIS terminology);
- The fourth column outlines the agents (according to PREMIS terminology) linked to the event.

Event	Value of the PREMIS eventType element	Values of the PREMIS linkingObjectIdentifier elements	Values of the PREMIS linkingAgentIdentifier elements
Creation of the archive	information package creation	• Repository code • Collection code • Lot identifier • Retention schedule code (optional)	• Login of the LDAP user • Arcsys
Submission of the archiving request	ingestion start	• Request identifier • Lot identifier • Repository code • Collection code • Retention schedule code (optional)	• Login of the LDAP user • Arcsys
Computation of the checksum of the objects	message digest calculation	• Lot identifier • Object identifier	• Arcsys Application Agent
Digital signature generation	digital signature generation	• Lot identifier • Object identifier	• Arcsys Application Agent
Format identification, validation and signature validation	format identification	• Lot identifier • Object identifier	• Arcsys Application Agent
End status of archiving request (successful, error or cancellation)	ingestion end	• Request identifier • Lot identifier	• Arcsys Engine

Event	Value of the PREMIS eventType element	Values of the PREMIS linkingObjectIdentifier elements	Values of the PREMIS linkingAgentIdentifier elements
High-level archive modification	metadata modification	• Lot identifier • Class path (optional)	• Login of the LDAP user • Arcsys
Document fixity check (error)	fixity check	• Lot identifier • Object identifier	• Arcsys Application Agent
Native document fixity check (error)	fixity check	• Lot identifier	• Arcsys Application Agent
Envelope fixity check (error)	fixity check	• Lot identifier • Object identifier (if available)	• Arcsys
Permissions change	permissions change	• Class path for class permissions change • Collection code for collection permissions change • Lot identifier for lot permissions change • Repository code for repository permissions change	• Login of the LDAP user • Arcsys
Download/Retrieval	dissemination	• Lot identifier • List of object identifiers if download or partial retrieval	• Login of the LDAP user • Arcsys Engine
Archive restitution	restitution	• Lot identifier	• Login of the LDAP user • Arcsys Engine
Migration of an archive	refreshment	• Lot identifier • Request identifier	• Login of the LDAP user • Arcsys Engine
Copy of an archive	replication	• Lot identifier • Request identifier	• Login of the LDAP user • Arcsys Engine
Deletion of an archive (full deletion, or soft delete of objects in an archive)	deaccession	• Lot identifier • Login of the LDAP user (if soft delete) • Object identifier (if soft delete)	• Arcsys (if soft delete) • Arcsys Engine (if full deletion)

Event	Value of the PREMIS eventType element	Values of the PREMIS linkingObjectIdentifier elements	Values of the PREMIS linkingAgentIdentifier elements
		• Request identifier (if full deletion)	
Creation of a retention schedule	creation	• Repository code • Retention schedule code	• Login of the LDAP user • Arcsys
Modification of a retention schedule	modification	• Repository code • Retention schedule code	• Login of the LDAP user • Arcsys
Deletion of a retention schedule	deletion	• Repository code • Retention schedule code	• Login of the LDAP user • Arcsys
Creation of a class	creation	• Repository code • Class path	• Login of the LDAP user • Arcsys
Modification of a class	modification	• Repository code • Class path	• Login of the LDAP user • Arcsys
Deletion of a class	deletion	• Repository code • Class path	• Login of the LDAP user • Arcsys
Attaching a policy to a class	policy assignment	• Repository code • Class path	• Login of the LDAP user • Arcsys
Assigning a retention schedule to an archive	policy assignment	• Lot identifier	• Login of the LDAP user • Arcsys
Creation of a hold	creation	• Lot identifier	• Login of the LDAP user • Arcsys
Deletion of a hold	deletion	• Repository code • Class path	• Login of the LDAP user • Arcsys
Modification of a metadata (regarding an archived lot)	metadata modification	• Collection code • Lot identifier • Object identifier (in the case of metadata at document level)	• Login of the LDAP user • Arcsys

Event	Value of the PREMIS eventType element	Values of the PREMIS linkingObjectIdentifier elements	Values of the PREMIS linkingAgentIdentifier elements
Journal generation	journal generation		• Arcsys Engine

Table 4.1. List of events generated in journals

The XML events contain JSON information about the detail and the outcome of the operations. The following table lists the names of the JSON files of the `jsonSchemas` subfolder of the `doc` folder of the packaging.

Event	Detail JSON files	Outcome JSON files
Creation of the archive	`lot-creation-detail.json` (extends `simple-entity-detail.json`, permissions in `lot-permission.json`)	`success-outcome.json`
Submission of the archiving request	`request-archiving-detail.json` (extends `request-detail.json` which extends `simple-entity-detail.json`)	`success-outcome.json`
Computation of the checksum of the objects	`object-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json`
Digital signature generation	`signature-detail.json` (extends `object-detail.son`)	`success-outcome.json`
Format identification, validation and signature validation	`format-identification-detail.json` (extends `object-detail.json`)	`format-identification-outcome.json`
End status of archiving request (successful, error or cancellation)	`request-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json` in case of success, `request-outcome.json` in case of error
High-level archive modification	`lot-modification-detail.json` (extends `simple-entity-detail.json`, content in `lot-modification-content.json`)	`lot-modification-outcome.json` (content in `lot-modification-content.json`)
Document fixity check (error)	`object-detail.json` (extends `simple-entity-detail.json`)	`object-fixity-error-outcome.json` (extends `success-outcome.json`)
Native document fixity check (error)	`native-object-detail.json` (extends `simple-entity-detail.json`)	`native-object-fixity-error-outcome.json` (extends `success-outcome.json`)
Envelope fixity check (error)	`mover-detail.json` (extends `simple-entity-detail.json`)	`mover-fixity-error-outcome.json` (extends `success-outcome.json`)
Permissions change	`permission-detail.json` (extends `simple-entity-detail.json`, content in `permission.json`)	`permission-outcome.json` (content in `permission.json`)
Download/Retrieval	`request-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json`

Event	Detail JSON files	Outcome JSON files
Migration of an archive	`request-migration-detail.json` (extends `request-detail.json` which extends `simple-entity-detail.json`)	`success-outcome.json` in case of success, `request-outcome.json` in case of error
Copy of an archive	`request-migration-detail.json` (extends `request-detail.json` which extends `simple-entity-detail.json`)	`success-outcome.json` in case of success, `request-outcome.json` in case of error
Deletion of an archive (full deletion, or soft delete of objects in an archive)	Full deletion: `request-detail.json` (extends `simple-entity-detail.json`) ; soft delete: `soft-delete-detail.json`	Full deletion: `success-outcome.json` in case of success, `request-outcome.json` in case of error; soft delete: `success-outcome.json`
Creation of a retention schedule	`retentionSchedule-creation-detail.json` (extends `simple-entity-detail.json`, start trigger in `retentionStartTrigger-entity.json`, end trigger in `retentionEndTrigger-entity.json`)	`success-outcome.json`
Modification of a retention schedule	`retentionSchedule-modification-detail.json` (extends `simple-entity-detail.json`, content in `retentionSchedule-modification-content.json`)	`retentionSchedule-modification-outcome.json` (content in `class-retentionSchedule-modification-content.json`)
Deletion of a retention schedule	`simple-entity-detail.json`	`success-outcome.json`
Creation of a class	`class-operation-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json`
Modification of a class	`class-modification-detail.json` (extends `class-operation-detail.json`, content in `class-modification-content.json`)	`class-modification-outcome.json` (content in `class-modification-content.json`)
Deletion of a class	`class-operation-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json`
Attaching a policy to a class	`policy-attachment-detail.json` (extends `simple-entity-detail.json`, content in `policy-assignment-type.json`)	`success-outcome.json`
Assigning a retention schedule to an archive	`policy-attachment-detail.json` (extends `simple-entity-detail.json`, content in `policy-assignment-type.json`)	`success-outcome.json`
Creation of a hold	`hold-creation-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json`

Event	Detail JSON files	Outcome JSON files
Deletion of a hold	`hold-deletion-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json`
Modification of a metadata	`metadata-modification-detail.json` (extends `simple-entity-detail.json`)	`success-outcome.json`
Journal generation	`journal-generation-detail.json`	`journal-generation-outcome.json`

Table 4.2. JSON files for the events

4.2. Entities

Each event contains a JSON field `entity` within the `eventDetailInformation` section. This field represents the entity type according to the data model of Arcsys from which the data was extracted to log an event.

This field can have the following values:

- `repository`: events of granting and modifying permissions on a repository;
- `collection`: events of granting and modifying permissions on a collection;
- `class`: events resulting from interactions with classes such as creation, modification, deletion, assignment of a policy (format, attestation, or workflow), and assignment of a retention schedule;
- `archive`: events resulting from interactions with lots (excluding requests) such as creation, high-level modification, assignment of a retention schedule or a class, modification of permissions and modification of metadata (on archived lots);
- `document`: events resulting from interactions with objects such as checksum calculation, checksum verification errors, format identification, signature, deletion on an archived lot (soft delete) and modification of metadata (on archived lots);
- `retentionSchedule`: events of creation, modification, and deletion of a retention schedule;
- `disposalHold`: events of creation and deletion of a disposal hold;
- `request`: events related to the lifecycle stages of a request such as the submission of an archiving request, the completion of an archiving, retrieval, restitution, deletion, migration, copy, or display request (terminal status, whether successful or not), the cancellation of a request of these types (except display);
- `envelope`: events of checksum verification errors reported by the Arcsys Transfer Server.

Part 12. Proof slip and proof folder

1. Definition of a proof slip

1.1. Introduction

The proof slip is a PDF document which provides as much information as possible on an archive or an object archived in Arcsys. It associates the original document inseparably with the sequence of events occurring to the document in the system. Its aim is to demonstrate the fixity and authenticity of the documents for admission as proof.

1.2. Proof slip Format

The proof slip is a document in PDF format that contains 6 parts, organized in a table:

- Sources of the Proof slip: Section that lists the different sources used to establish the proof slip;
- Document identification: Section that lists the information that allows the document to be clearly identified;
- Document indexing: Section that lists the indexing information (metadata) and classification information (classification scheme) for the document;
- Document archiving context: Section that lists the archiving information of the document (Repository, Collection, storage policy, Permissions);
- Document accountability: Section that lists information which allows establish the indisputable accountability of the document;
- Document life cycle: Section that lists the different Arcsys requests on the document and/or its lot (archiving, synchronous retrieval, migration, copy, retrieval, archive restitution, etc). It also lists the holds created or deleted on the archive.

1.3. Definition of proof folder

The proof folder includes the proof slip along with:

- The selected archive or documents;
- Additional items such as the XML common signature file or the timestamp response.

1.4. Proof slip and proof folder generation

The **proof slip** can be generated on archive or lot level:

- On the web interface (Arcsys Web Agent or ArcWeb Module);

- Using ArcBWS APIs of Arcsys REST API: see **Arcsys API Manual** for more details.

As the **proof folder** generation requires retrieving document with an elaborate demand, the proof folder can only be generated with ArcEP. See the manual of this product for more details.

1.5. Authenticity of the proof slip

At the end of the proof slip, the following sentence is written: `The digital fingerprint of this proof slip is notified in the log file of the generation module.` This allows verification that the proof slip has not been altered after its generation. The "generation module" may be either Arcsys Web Agent, or Arcsys REST API (in the case of a proof folder generation with ArcEP, the proof slip is generated by Arcsys REST API).

The digital fingerprint is recorded in the following log line:

- `Proof slip for archive ... document ... has hash value: ...` for an archive-level proof slip
- `Proof slip for archive ... has hash value: ...` for a document-level proof slip

Part 13. Business and Technical Domains

1. Introduction

With Arcsys, you can distinguish lots, objects and keywords by the business or technical domain. Technical lots, objects and keywords are not meaningful from the end-user's point of view; they exist only for internal Arcsys operation and its relevant connectors.

The aim is to avoid putting mandatory metadata (business data) on objects that do not require it.

Here are some examples of use:

- Trace type lots (for the collection level traces, not the repository level traces) are technical lots;
- The `SIP.xml` and `AIP.xml` files are stored in technical objects in the case of ArcIP;

2. Functional Rules

Certain functional rules are applied concerning the concordance of business and technical domains between lots, objects and keywords:

- A business lot can have business and technical objects;
- A business lot can have metadata for business and technical keywords;
- A business object can only belong to a business lot;
- A business object can have metadata for business and technical keywords;
- A technical lot can only have technical objects;
- A technical lot can only have metadata for technical keywords;
- A technical object can belong to a business or technical lot;
- A technical object can have metadata for technical keywords only.

Important

Technical lots will systematically have their source files deleted at the end of archiving.

Part 14. Keyword and Metadata Configuration

1. Defining Keywords and Metadata

1.1. Concepts and Definitions

A **keyword** defines a field in which metadata can be entered. It is a configurable element, designed to capture specific information about an archive or document.

A **metadata** is a value assigned to a keyword for a particular archive or document. In other words, keywords are the structure, while metadata are the actual values stored and indexed.

For example, "Author" can be a keyword. The metadata for this keyword could be "Jane Doe" for one archive, and "John Smith" for another.

1.2. Keyword Types

A keyword can belong to one of four types:

- String
- Numerical
- Date
- Controlled (list of values)

Some types also support subtypes:

String subtypes:

- None
- Extended date

Date subtypes:

- None
- Date with hour

1.3. Organizing Elements in Masks

A mask defines which metadata are relevant in a given context by selecting the appropriate keywords to be used. It also controls how they are presented in the user interface, including their order, grouping, and whether they are mandatory.

It consists of an ordered list of elements. These elements can be either keywords or visual separators, described in more detail in the following section.

Each element added to a mask is associated with the following specific properties:

- *Order:* Defines the position in the mask.
- *Level:* "Lot" or "Object", indicating the applicable context.
- *Domain:* "Business" or "Technical", used to organize the display.

In addition, for keywords only:

- *Mandatory:* Indicates whether the keyword requires a value during metadata entry.
- *Modifiable:* Indicates whether the value of the keyword can be modified after a lot has been archived. By default, this behavior is enabled. If disabled, it is no longer possible to add, modify, or delete the value of the keyword once the lot is archived. Only a user with the **GRANT_ALL** role is allowed to override this restriction.

These properties belong to the association between the element and the mask — not to the element itself. A keyword or separator can be used multiple times across different masks, with different values for these properties.

1.4. Keyword Separators

A *separator* is a visual element used in masks to group keywords into sections (e.g., "Identity", "Address"). Separators improve clarity without storing or indexing any data.

Each separator has a *code* and a *label*, defined at the repository level. The same separator can be reused in multiple masks.

A default separator named `Arcsys_Blank_Separator` is always available.

1.4.1. Creating, Editing and Deleting Separators

You may:

- *Create:* by providing a code and label during mask editing.
- *Edit:* only the label, which affects all uses of that separator.
- *Delete:* only if the separator is not used in any mask.

1.5. Display and Ordering Rules

1.5.1. Ordering in Masks

Keywords and separators share a common ordered list. This order defines how metadata entry and display forms are structured for the user.

Elements from different levels (lot and object) can be mixed in the same mask.

1.5.2. Display Behavior in Different Contexts

The display of a mask depends on the usage context:

- **Lot creation or update:** Only lot-level keywords and separators are shown, in order.
- **Object creation or enrichment:** Only object-level elements are shown.
- **Viewing or editing an object:** All elements from both levels are shown, respecting the defined order.
- **Search screen:**
  - **Single mask:** The full structure, including separators and defined order, is preserved.
  - **Multiple masks:** Separators are omitted. Keywords are grouped by level (lot first, then object), and the order is computed cumulatively:
    - Start with the first mask.
    - Add any new keywords from other masks, preserving their relative order.

2. Internal and External Indexing

2.1. Indexing Policy

The **indexing policy** defines whether metadata and/or document content should be indexed internally only, or also externally in a third-party search engine (referred to here as *GenericSearch*).

Important

External indexing requires ArcRFT Option.

- **Metadata:** Always indexed in Arcsys Database. Optionally, also in GenericSearch.
- **Document content:** Not indexed by default. Can be indexed in GenericSearch if configured.

2.2. Indexing Process in GenericSearch

When external indexing is enabled, the following elements are indexed:

- Repository, Collection, Lot, Document identifiers and codes
- Lot creator
- Document path
- Class code
- Metadata (each keyword=value association)

These are sent:

- From Arcsys Web Agent, during archive preparation or metadata editing
- From Arcsys Engine, in `INITIALIZED` status

If document content indexing is enabled, documents are converted using Apache Tika and sent to GenericSearch during `VALIDATED` status.

2.3. Deleting Indexed Information

Information is removed from GenericSearch in the following cases:

- Document content: upon deletion of the document or final migration of the last copy
- Metadata and common characteristics: after lot or document deletion

- All indexed data: if an error occurs during initial archiving

Part 15. Encrypting data in Arcsys: concepts

1. Introduction

Encryption may be used in Arcsys for:

- encrypting data *in motion* during the data transfer operation performed by the components of its architecture;
- storing data *at rest* on storage (filesystems, tapes) managed by ArcMover. This is the goal of ArcCrypt Option.

2. Encrypting data in motion with SSL

All data transfer protocols used in Arcsys (RMI, HTTP, LDAP, JDBC, TCP/IP) can be encrypted with SSL with an appropriate configuration. See **Arcsys Administration Manual** for more details.

3. Encrypting data at rest with ArcCrypt Option

Encryption and decryption are configured on a collection level for all storage pools. More precisely, the administrator chooses an **encryption policy** on the collection, which determines the encryption characteristics.

When encryption is enabled, the data and the XML manifest content are encrypted on all copies. An encryption policy is mainly characterized by the choice of a **master key** and an encryption algorithm.

The master key (stored in a PKCS#12 file or read through PKCS#11 interface in a Hardware Security Module, or HSM) is used to encrypt a *Collection Encryption Key* (that we will call "CEK"). This master key is read through a **Key Manager** configured in the Arcsys Transfer Server.

The encryption of the CEK is done with an algorithm configured in the master key. It is strongly advised to use an **asymmetric algorithm** (RSA2048 is supported).

This CEK is unique per collection and is used to encrypt all the archives of the collection with a symmetric algorithm.

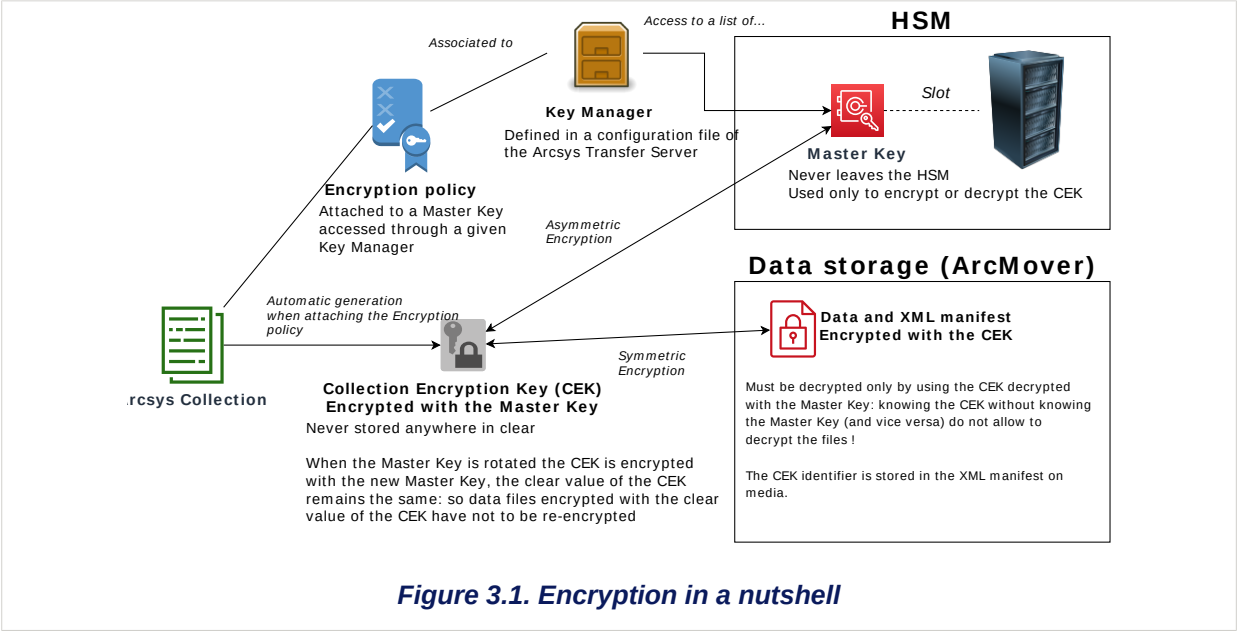

Figure 3.1. Encryption in a nutshell

Important

Encryption can be defined on a collection even if non encrypted archives are already present in it. However, please note that the encryption will only apply to new writing operations on archives of

this collection (new archives and migrations, recoveries, copies or synchronizations of exixting ones). The encryption policy cannot be removed from the collection when defined on it and applied to archives.

4. Master Key rotation

The ArcCrypt Option provides encryption for data at rest. The **Master Key rotation** feature lets administrators replace an existing Master Key with a new one while keeping all stored data valid.

4.1. Concept and purpose

This feature is based on the dual-key architecture used in Arcsys, which combines a *Master Key* and a *Collection Encryption Key (CEK)*.

The Master Key encrypts the CEK, and the CEK encrypts the archived data. When a rotation occurs, only the CEKs are re-encrypted with the new Master Key. The encrypted data stored on media remains unchanged. This design lets the Master Key be renewed without rewriting or re-encrypting the archived content. It makes the operation faster and lighter than a full re-encryption of stored data.

Re-encrypting many CEKs can still take time. To keep the system responsive, the rotation runs asynchronously through a deferred request. This ensures continuous service availability while encryption references are updated in the background.

During a rotation, all operations on the affected collections stay available. Users can still browse, export, migrate, and archive data normally.

A Master Key can be marked as *deprecated* to prevent its use in new encryption policies. A deprecated key remains valid for existing policies and for reading or writing existing archives. It is no longer selectable when defining a new encryption policy. When a rotation completes, the replaced key automatically becomes deprecated. A deprecated key can later be marked as active again if needed.

4.2. Operation

The rotation runs as a *deferred request* of type *Rotating master key*. The Arcsys Engine monitors and manages the process. It requests the Arcsys Transfer Server to re-encrypt each CEK with the new Master Key, then updates the related data in the database.

Rotation is possible only if several Master Keys are defined in the Key Managers. These keys can use different encryption algorithms, allowing the rotation to change the algorithm used for CEK encryption if needed. It is not possible to rotate toward a deprecated Master Key.

During processing, the system re-encrypts the *Collection Encryption Keys (CEK)* with the new Master Key and updates the related encryption policies. Only one rotation can run for a given Master Key at a time: a new rotation cannot be created while another is still running on the same Master key.

4.3. Triggering a Master Key rotation

The rotation is not automatic. It must be started manually by the administrator, either through ArcWeb Module or through the REST API of Arcsys REST API.

When triggered, the system creates a *deferred request* of type *Rotating master key*. This request is not associated with any specific repository. It can be monitored to track its progress until it ends, at which point the current Master Key becomes *deprecated*.

4.3.1. Triggering a rotation with graphical interface in ArcWeb Module

In ArcWeb Module, open *Functional administration* > *Security* > *Master keys*. The page lists all available Master Keys and their details. Each key entry provides management actions.

- **Rotate master key** starts the rotation process for the selected key
- **Mark as Deprecated** makes a key unavailable for new encryption policies without triggering a rotation

When selecting *Rotate master key*, a dialog appears. The administrator chooses the new Master Key that replaces the current one, which becomes *deprecated* once the rotation request has been processed.

Note

The key list can include or exclude deprecated keys by using the *Include deprecated master keys* filter.

4.3.2. Triggering a rotation with REST API endpoint

A rotation can also be triggered through the REST API. Use the following **PATCH** request to start the rotation and specify the new Master Key identifier:

```
PATCH /api/masterKeys/{masterKeyId}/_rotate {"newMasterKeyId":"xxx"}
```

The response returns a JSON object (`api-masterKey-defer-request.json`) that contains the characteristics of the created request.

4.4. Deferred request processing

When a *Rotating master key* request is detected by the Arcsys Engine, the following operations are performed:

1. The Arcsys Engine identifies all encryption policies that use the source Master Key
2. For each policy, it retrieves all CEKs linked to it
3. The Arcsys Transfer Server re-encrypts each CEK using the new Master Key and the algorithm defined in that key
4. The Arcsys Engine updates the encrypted CEKs and the related encryption policies in the database
5. All updates are committed in one transaction to ensure data consistency
6. The previous Master Key is automatically marked as deprecated when the rotation completes

The processing of a deferred request can end with two statuses:

- `ENDED`: the rotation completes successfully and all encryption policies are updated with the new Master Key
- `ERROR`: the rotation stops before completion. Some encryption policies may still reference the old Master Key and their CEKs may not have been re-encrypted

If a rotation request ends in `ERROR`, the administrator must manually create a new rotation request. This new request automatically targets only the encryption policies that still reference the previous Master Key.

Important

The previous Master Key must not be deleted from the HSM or keystore until the rotation process is fully completed. The key remains required to read data that still uses CEKs encrypted with it during the rotation phase.

4.5. Configuration and prerequisites

At least two Master Keys must be defined to run a rotation. Master Keys are managed by *Key Managers*, which are configured in the file defined by the *`key_managers_conf`* parameter in the `etc/transferServer.conf` file of the Arcsys Transfer Server.

Each Transfer Server is linked to one `key_managers_conf` file. This configuration file can reference several keystores. Supported keystore types are:

- `PKCS#12`: each keystore contains a single Master Key, and each Key Manager entry in the configuration file corresponds to one keystore
- `PKCS#11`: one keystore can contain several Master Keys

For details about defining and configuring Key Managers, see Arcsys Administration Manual.

4.6. Permissions and roles

Operations related to Master Keys and encryption management (including rotation, creation, and deprecation) require the `ENCRYPTION_MANAGEMENT` role in LDAP.

Part 16. Modifying an archived lot

1. Introduction

In Arcsys, it is possible to perform addition and deletion operations on objects within lots that have already been archived. However, directly modifying the objects within a lot after they have been archived is not possible.

2. Deleting an object

2.1. Description of the feature

The deletion of an object from an archived lot can be done through the Arcsys Web Agent, the Arcsys REST API or ArcWeb Module, in the same way as deleting an object from a lot that is not archived.

Unlike deleting an object from a lot that is not archived, it does not delete the object from the database or delete the data on the media. The object information is still visible in the interface, but the object is flagged as deleted in the database via a special column.

A deleted object can no longer be retrieved or downloaded but it can still be used for integrity checks.

The deletion of all the objects in an archived lot results in the generation of a disposal request for the lot in question.

Note

It is not possible to delete an object from an held lot.

2.2. Rules

If a user does not have **GRANT_ALL** role, he must have the **OBJ_DELETE** role to delete an object from an archived lot.

3. Adding an object

3.1. Description of the feature

Adding an object to an archived lot can be done through the Arcsys REST API or ArcWeb Module using lot enrichment requests.

The enrichment of a lot occurs in two steps:

- The creation of one or more enrichment type objects in an archived lot;
- The creation of an enrichment type request on the lot, that will perform archiving operations on these various objects.

The enrichment requests follow the same life cycle as a typical archiving request, aiming to perform the same operations without modifying the previously archived data and, consequently, the existing envelopes. Thus, enriching a lot will inevitably result in the creation of at least one new envelope.

During a lot enrichment, archival properties used during lot archiving, such as compression, encryption, maximum envelope size, and various properties defined at the class level (format policies, indexing policies, workflow policies, attestation policies, etc.), are not reused. These properties are defined by the configuration at the time of enrichment. Thus, after enrichment, a lot may end up with envelopes having different properties.

Important

Enrichment does not support timestamping and signing agents. All configuration related to these aspects is ignored during enrichment.

3.2. Rules

If a user does not have **GRANT_ALL** role, he must have the **LOT_ENRICHMENT** role to:

- Create an enrichment object in an archived lot;
- Delete an enrichment object from an archived lot;
- Create an enrichment request.

3.3. Arcsys REST API endpoints

As mentioned earlier, enrichment takes place in two steps. The Arcsys REST API endpoints enabling these two operations are:

- `/lots/{lotId}/objects/_addObject`: Creates an enrichment object in an archived lot;
- `/lots/{lotId}/requests/_enrich`: Creates an enrichment request on an archived lot.

4. Moving a lot

4.1. Description of the feature

Moving an archived lot from one collection to another can be done through the Arcsys REST API or ArcWeb Module.

4.2. Rules

4.2.1. Rights

If a user does not have a **GRANT_ALL** role, he must have the **ARCHIVE** and **PREPARE_ARCHIVE_OBJ** roles to move the lot from one collection to another.

4.2.2. Business

The following business rules must be respected to move a lot from one collection to another:

- The lot must be archived
- The lot must not be encrypted
- The target collection must not have encryption enabled
- The target collection must belong to the same repository as the source collection
- The target collection must have to the same storage profile as the source collection
- The target collection must either have the same mask as the source collection or a mask that contains all the keywords present in the source collection's mask, without containing additional mandatory keywords that have no default value.

4.3. Arcsys REST API endpoint

The Arcsys REST API endpoint that allows the lot move operation is `/lots/{lotId}/_move`.

Part 17. Transit zones

1. Role and definition

Archiving or retrieving data requires to choose an Arcsys Application Agent on which the archiving or retrieving operation will be performed. This concept can be tricky to comprehend for an end-user.

For this reason, the concept of transit zones is introduced: they provide a better way to define the Arcsys Application Agent to perform archiving and retrieval without needing a technical understanding of Arcsys or the underlying architecture.

As defined in the Glossary, a transit zone is a logical entity linking together an Arcsys Application Agent and a directory, as well as additional configurations.

Note

The name "zone" has been purposefully chosen for its closeness with the concept of Storage Zone.

2. Characteristics

A transit zone is a global level entity, thus not attached to a repository.

The main characteristic of a transit zone is the **Arcsys Application Agent** attached to it.

A transit zone has also the following characteristics:

- A code (required)
- A label (required)
- A path to the root directory to be used (required). For example : `/mnt/users`. The user will fill in a relative path to that root path. This root path must be an absolute path. The path may or may not have the trailing slash.
- A rule which may contain wildcards (following "glob" syntax) to define the allowed documents and directories in the root directory.

  For example if you define `u00*/*` for the rule, and the root directory is `/mnt/users` , that means that archiving files within `/mnt/users/u0042/test1` are authorized, but files named `/mnt/users/test` are forbidden. If you also want to allow archiving files within `/mnt/users/u0042/test1/subfolder1` the rule should become `u00*/**`.

Note

For more details on the "glob" syntax see for example https://en.wikipedia.org/wiki/Glob_(programming)

Important

Please note that this rule cannot be used to restrict archiving to folders only: the rule cannot end with \ or /.

- A flag defining whether archiving is allowed on this zone (default `true`)
- A flag defining whether retrieval is allowed on this zone (default `true`)

3. Rules

3.1. Rights

Creating, modifying or deleting a transit zone requires to have the **ADMIN** role.

3.2. Business rules

Modifying or deleting a transit zone is possible at any time and takes effect as early as possible (when the next request impacted is created). A request in progress will not be affected by the change.

It is not possible to delete an Arcsys Application Agent if it is attached to a transit zone.

4. Using transit zones

Transit zones:

- Can be created, modified and deleted through dedicated operations in Arcsys REST API;
- Can be used to perform archiving operations without choosing a Arcsys Application Agent, with a dedicated operation in Arcsys REST API, and in the "advanced archiving" feature of ArcWeb Module.

Note

Transit zones are not accessible in Arcsys Web Agent.

Part 18. Aggregate feature

1. Description

Important

This chapter refers to a feature preview (see Feature preview for more details).

This feature aims to provide metrics about the requests in Arcsys. The idea behind aggregation is straightforward: rather than performing extensive calculations on existing data (like summing numbers) all at once, we choose to process and prepare the data in advance, making it ready to use whenever we need it. This approach makes the data easier and quicker to work with.

The aggregation is called each time a request is ended. It is not retroactive: the past requests are not considered.

The aggregation does not allow us to keep a record of each request individually, but rather of a group of requests defined by the same properties. The data generated by the aggregation is meant to be used in statistics and dashboards: this is where the reporting comes in.

Note

This feature is not activated by default in Arcsys.

2. Available data

Important

This chapter refers to a feature preview (see Feature preview for more details).

The purpose of the aggregate table is to monitor the number, size, and status of completed requests made in Arcsys, along with the number of objects associated with these requests.

This information is organized along a temporal axis, with each value linked to the appropriate:

- Repository
- Collection or Class
- Request type
- Engine

From this aggregate, users can select the necessary data.

The aggregation will be called each time a request is completed in Arcsys, in the engine for most requests and in the Arcsys REST API for the Synchronous Retrievals. Based on the parameters, it will update the information in the database.

The update is in real-time and occurs almost instantly once the request is completed.

2.1. Field details

The following fields are described to enable the user to query the aggregate table for reporting purposes, but Arcsys does not support any alterations to this data. If the user modifies or deletes data in the database, they risk rendering it irrelevant. The aggregate contains the following fields:

```
AGR_STARTDATE | AGR_ENDDATE | AGR_BASIDENT | AGR_CLA_ID | AGR_COLIDENT | AGR_REQTYPE |
AGR_ENGIDENT | AGR_NBREQ | AGR_SZREQ | AGR_NBREQERROR | AGR_NBOBJ ;
```

These fields are divided into two types: **context fields** that define the context of this data, and **data fields**, which hold the actual data.

2.1.1. Context Fields

AGR_STARTDATE and AGR_ENDDATE define the time range for which the data is available. These two columns are stored in Date + Time format.

AGR_BASIDENT: numerical identifier, refers to the repository linked to the request.

AGR_CLA_ID: numerical identifier, refers to the class linked to the archive of the request (if there is a class). This field may not always be filled in, as the aggregate is made by collection OR class.

AGR_COLIDENT: numerical identifier, refers to the collection linked to the request. This field may not always be filled in, as the aggregate is made by collection OR class.

AGR_REQTYPE: single character representing the request type. The possible request types are:

- 'A': Archiving
- 'R': Retrieval
- 'O': Synchronous Retrieval
- 'M': Migration
- 'C': Check
- 'S': Deletion
- 'P': Recovery
- 'D': Copy
- 'Y': Synchronization
- 'U': Archive Restitution
- 'B': Lot Enrichment

AGR_ENGIDENT: numerical identifier, refers to the engine that processed the request. This field may not always be filled in, as some requests are not processed by an engine (e.g., synchronous retrieval/downloads).

2.1.2. Data Fields

AGR_NBREQ: numerical value representing how many times a request of type AGR_REQTYPE was completed, in success or in error.

AGR_SZREQ: numerical value representing the size in Kilobytes associated with the requests.

AGR_NBREQERROR: numerical value representing how many times a request of type AGR_REQTYPE ended in error.

AGR_NBOBJ: numerical value representing how many objects are impacted by the requests.

2.2. Aggregation

Below is an example of how the table will/should be filled in with a few requests.

Each line presents a unique combination of context fields. When aggregating the data, the line with the corresponding data will be either created (the first time this combination is used) or updated (the combination is already used, so Arcsys updates the data fields).

Example 2.1. Aggregation

If the reporting parameters are configured to track every request daily, and in Arcsys we have the following requests:

- 100 Synchronous Retrieval Requests, on collection 5, repository 4 on the 01/01
- 40 Archiving requests, on collection 5, repository 4 on the 01/01
- 50 Archiving requests, on collection 6, class 3 and repository 4 on the 01/01
- 70 Archiving requests, on collection 5, repository 4 on the 02/01. 32 requests are handled by engine 1, 38 are handled by engine 2.

Then you would obtain the following results in the aggregate table:

Start date	End date	Reposi-tory Id	Class Id	Colle-ction Id	Engine Id	Request type	Number of requests	Number of requests in error	Size of requests	Number of objects
01/01 0h00	02/01 0h00	4		5		'O'	100	2	10020	196
01/01 0h00	02/01 0h00	4		5	2	'A'	40	0	4002	75
01/01 0h00	02/01 0h00	4		6	1	'A'	50	0	5001	98
01/01 0h00	02/01 0h00	4	3		1	'A'	50	0	501	98
02/01 0h00	03/01 0h00	4		5	1	'A'	32	0	3203	32
02/01 0h00	03/01 0h00	4		5	2	'A'	38	0	4804	48

Each of these groups of requests creates its own line in the database. Indeed, they all have unique combinations. Note the special case of the requests that are both in class

and collection: the data is present twice in the table. It is possible to have the detail per collection OR by class, but not both at the same time.

3. Data Access

Important

This chapter refers to a feature preview (see Feature preview for more details).

You can access this data from the database directly. Simply make SQL requests to the corresponding table: AGR_AGGREGATE_REPORTING.

To customize the results, you can filter on some columns (ex: to obtain the data corresponding to a single repository), and do partial aggregation on others (to abstract/ignore some fields;ex: if we only want the aggregate to be made per repository, we need to abstract all the other fields).

- To filter the data, use a `WHERE` clause. You can filter both on 'context' and 'data' fields. For example: if you want the total volume of archiving requests in the repository X, you must add a `WHERE AGR_BASIDENT=X` clause.
- To obtain a partial aggregation, use the `SUM` and the `GROUP BY` statements. This can be used for 'context' fields. It can be done via the `SUM` and the `GROUP BY` statements. For example: you want the detail of the number of requests per Engine via a `SELECT SUM(AGR_NBREQ) FROM … GROUP BY AGR_ENGIDENT`.

For instance, if you wish to only obtain the results for the repository 1, and keep the detail of class,collection and request type, an appropriate request would be:

```
SELECT
      -- The Context fields that we want to keep
      AGR_STARTDATE, AGR_ENDDATE, AGR_BASIDENT, AGR_CLA_ID, AGR_COLIDENT, AGR_REQTYPE,

      -- The Data fields that we want to aggregate
      SUM(AGR_NBREQ) as "Number of requests",
      SUM(AGR_SZREQ) as "Size of requests",
      SUM(AGR_NBREQERROR) as "Number of requests in error",
      SUM(AGR_NBOBJ)  as "Number of objects"
FROM AGR_AGGREGATE_REPORTING
WHERE AGR_BASIDENT=1
GROUP BY
      -- The Context fields that we want to keep
      AGR_STARTDATE, AGR_ENDDATE, AGR_BASIDENT, AGR_CLA_ID, AGR_COLIDENT, AGR_REQTYPE;
```

Important

A request impacting both class and collection will be represented twice in the database. For any aggregate that does not imply this level, it is important to only aggregate the lines with a collection.

This approach is the best because a repository always contains collections, but not always classes.

4. Feature activation

Important

This chapter refers to a feature preview (see Feature preview for more details).

To enable aggregation in Arcsys, you must update the corresponding parameters. Modifying these parameters is only supported via the Arcsys REST API (either directly or through Swagger).

1. **Identify the requirements**

   Before enabling aggregation, it is recommended to identify your requirements. Determine what data will be the most useful for you, as the scope of the aggregation depends on your specific needs. It is possible to activate the aggregate for only a few repositories and request types, but also to define the time precision.

   Optimal parameters will reach the balance between having all the data you need and not taking up significant storage space on the database system. It is especially important to setup the time interval correctly: once the aggregation is done, there is no way to further refine the data: for instance, to go from daily to hourly.

   The parameters choices are defined in a ‘Reporting Parameter’ object. In order to build such an object, you need to specify the repository and request type for the aggregation, as well as the time-based value for the interval. The time-based value is specified in minutes: for an hourly aggregation, specify 60 minutes; for a daily aggregation, specify 1440 minutes (60*24).

2. **Access to the APIs to activate or modify aggregation**

   The operations are available in the following resources:

   - /reportingParameters to update/delete a given parameter,
   - /repositories/{repositoryId}/reportingParameters to create/delete parameters in a repository.

Note

As for the Arcsys REST API, the Swagger documentation is the easiest way to master these operations.

3. **Apply the changes**

   You can initialize and change the parameters at any moment in Arcsys. To ensure the parameters are updated, you must restart the Arcsys REST APIs and the Arcsys Engines after any modification.

Important

Changing the time interval once aggregation has started may cause strange data results: overlapping time intervals. However, no data will be lost and it will still be exploitable.

For example, original aggregation is daily, and we decide to switch it to hourly mid-day, while thousands of archiving request are still running. We will have the original aggregate line with a time interval from midnight to midnight, and multiple hourly intervals from 12pm to midnight. Each request will still only be aggregated once, and we can query the data the same way we did before.

5. Data initialization

Important

This chapter refers to a feature preview (see Feature preview for more details).

The aggregation as described earlier does not allow the consideration of past requests. If you were already using Arcsys before installing the Aggregation Feature, and wish to have some insight from the data of the past, you can run the ReportingInitializer tool beforehand.

5.1. Setup

The batch tool runs on a specific set of parameters:

- `Repository codes`: (optional) Defines the repositories on which the batch will aggregate. If the parameter is left blank, all the repositories are aggregated, by default.
- `Request types`: (optional) Defines the types of requests on which the batch will aggregate If the parameter is left blank, all the request types are aggregated, by default.
- `Start date / End date`: Defines a time range for the batch to execute on. This parameter is mandatory. ReportingInitializer only aggregates the requests run between the two specified dates.
- `Time interval`: (optional) The time detail that will be in the aggregate:
  - Integer: number of minutes that will be used as time interval (`60`: hourly interval)
  - No value: No time detail will be used (Direct aggregation from start end to end date)

These parameters define the scope of data that needs to be aggregated. The simplest way to aggregate past data is to fill in the start date and end date parameters: with the Arcsys installation date in start date, and the activation of the aggregate feature date in the end date, all your Arcsys history will be aggregated.

Important

Be careful if you wish to modify the scope of the batch and/or run multiple instances. Data inconsistencies can be easily produced, either by aggregating twice on the same scope, or by missing entire parts of the history.

5.2. Processing

Upon execution, the program initially displays key metrics related to the selected data and then pauses, waiting for user confirmation before proceeding with the aggregation. Processing might take a while, depending on the size of the request history, but also on the scope and time granularity defined in the parameters.

6. Impacts on performances

Important

This chapter refers to a feature preview (see Feature preview for more details).

In test environments, there is no significant impact in terms of processing time.

However, the volume of data can become substantial, depending on the chosen parameters. To calculate the maximum possible volume depending on the parameters, you have to calculate the number of possible values in the database. For each combination of repositories and request type, you can select a different time interval. The maximum number of lines added for each time interval can be calculated by multiplying:

- the addition of the number of classes and the number of collections in the activated repositorie
- the number of activated request types
- the number of engines

You can then calculate the maximum volume by multiplying the maximum number of entries per time interval, the volume of a line, and the number of times the time interval will be represented over a period (typically, a daily interval over a year will represent 365 possible time values). Each aggregated line is about 100 bytes.

For reference, using aggregation for 10 years with 20 different collections and classes, 5 engines for all request types (13), and daily detail would never represent more than 237.25 MB in the database. (20*5*13 = 1300 lines maximum per time interval, 10*365 = 7300 different time intervals, 100 being the size of the line, 325*7300*100 = 237 250 000 bytes = 237.25 MB)

This volume only represents the upper bound of the volume, and depending on how Arcsys is used, it could be significantly less.

Part 19. Access zones

1. Definition

Access zones are used to enforce security policies by restricting access to resources based on the requester’s network origin. An access zone is an independent entity within Arcsys that defines a controlled network area from which resources can be accessed. These entities can then be attached to permissions (at the repository, collection, lot, or class level) to restrict or grant access based on the client's IP address when authenticating to the Arcsys REST API, the Arcsys Web Agent or ArcWeb Module. An access zone is typically represented by a list of authorized IP addresses or IP ranges, designating the locations (such as offices, data centers, or trusted networks) where users are permitted to connect.

Important

The mechanism is implemented neither in the Arcsys RMI API nor in the Arcsys SOAP API. This means that any permission linked to an access zone will not be valid in these modules.

2. Characteristics

An access zone is a global level entity, thus not attached to a repository.

An access zone has also the following characteristics:

- A code (required);
- A label (required);
- A description (optional);
- A set of access restrictions (required), separated by commas. These restrictions can be:
  - Individual IP addresses (IPv4 or IPv6), e.g., `192.168.1.100` for IPv4 or `2001:db8::1` for IPv6;
  - IP address ranges (IPv4 or IPv6, format: <ip_address1>-<ip_address2>), e.g., `192.168.1.100-192.168.1.200` for IPv4 or `2001:db8::1-2001:db8::ff` for IPv6;
  - IP netmask in CIDR notation (IPv4 or IPv6, format: <ip_address>/<prefix_length>), e.g., `192.168.1.0/24` for IPv4 or `2001:db8::/64` for IPv6.

Note

These different types of access restrictions can be combined within the same access zone, including a mix of both IPv4 and IPv6 addresses. For example, an access zone could include an individual IPv4 address (`192.168.1.100`), an IPv6 address (`2001:db8::1`), an IPv4 IP range (`192.168.1.100-192.168.1.200`), and an IPv6 CIDR mask (`2001:db8::/64`), all separated by commas.

3. Rules

3.1. Rights

Creating, modifying or deleting an access zone requires to have the **ACCESS_ZONE_MANAGEMENT** role.

3.2. Business rules

It is not possible to delete an access zone if it is attached to a permission.

It is possible to modify or detach an access zone from a permission at any time, with the changes being applied for each new authentication to the Arcsys REST API, the Arcsys Web Agent or ArcWeb Module from that point onward.

4. Using access zones

Access zones:

- Can be created, modified and deleted through dedicated operations in Arcsys REST API or ArcWeb Module;
- Can be attached or detached from a permission (at repository, collection, lot or class level) through dedicated operations in Arcsys REST API or ArcWeb Module.

Note

Even though the access zone mechanism is considered by the Arcsys Web Agent during authentication, it is not possible to manage these entities or perform the above-mentioned operations within it.

Part 20. Notifications

1. Overview

Notifications are asynchronous messages generated by Arcsys and sent through predefined diffusion lists to inform users or administrators about specific events, actions, or requests requiring attention.

The creation of notification request is only available through the Arcsys REST API. Once created, notification request are processed by the Arcsys Engine as a *deferred request* of type *Notification*, ensuring reliable and asynchronous handling.

Note

Currently, only email-based notifications are supported. The configuration related to email sending from the Arcsys Engine is documented in the **Arcsys Administration Manual**.

2. Notification types

Different types of notifications exist depending on the context and purpose of the request:

- `Archival Request`: Triggered when a user requests archival rights for a given lot;
- `Access Request`: Triggered when a user requests access rights to download or retrieve a lot;
- `Assistance Request`: Triggered when a user submits a general assistance request, including a custom message.

3. Diffusion lists

A diffusion list defines the general context of a notification, including its usage type (Archival, Access, or Assistance) and its language (English or French).

Each diffusion list is identified by a unique code and a label.

Diffusion lists are associated with one or several recipients (see section below), which define who will receive the notifications.

Their creation, modification, and deletion are currently available only through the Arcsys REST API.

4. Recipients

A recipient represents a notification target.

Each recipient is identified by a unique code and label, and has a type (currently limited to `email`) along with a value. For recipients of type `email`, this value represents the email address.

Recipients can be associated with one or several diffusion lists to determine which notifications they will receive (see above).

Their management (creation, update, or deletion), as for diffusion lists, is currently available only through the Arcsys REST API.

Glossary

Access Zone

An access zone is an independent entity within Arcsys that defines a controlled network area from which resources can be accessed. These entities can then be attached to permissions (at the repository, collection, lot, or class level) to restrict or grant access based on the client's IP address when authenticating to the Arcsys REST API, the Arcsys Web Agent or ArcWeb Module.

API *(Application Programming Interface)*

The APIs provided by Arcsys enable the product holder to fully customize a new application or user interface according to the specific ergonomic needs of their use case. Arcsys exposes several types of APIs:

- REST APIs are the recommended interface. They offer broad coverage of Arcsys's functionalities, including administration, operations, archiving, search, and archive retrieval.
- Legacy APIs based on RMI and SOAP protocols are still available for compatibility purposes but are deprecated and should no longer be used in new developments.

Application Agent

There are two different types of agents at archiving level: application interface agents and user interface agents. An **application agent** can archive all the objects specific to an application (files, RDBMS table records, etc.), whereas a **web agent** performs both administration functions and manual archiving functions initiated by the user.

Archiving By Reference

Archiving by reference is a method in which data remains in its original storage location when added to an archive system, and the system generates references and metadata entries for the files. Eventually, the files are transferred to the archive system's defined storage using the copy and migration mechanism.

Archive Restitution

Archive restitution is the return and transfer of archived documents to their originator, or to a duly appointed person or organization. An Archive Restitution is in Arcsys an Archive Retrieval operation that ends with a Destruction. An Archive restitution operation can only be created through the appropriate operation in the REST API, or by using ArcEP module.
See Also **Archive Retrieval**, **Destruction**.

Archive Retrieval

Archive retrieval is an operation that makes a copy of a record available to a record requester. This term takes precedence over the term *restore*, which has another meaning at archiving level (restore in the sense of handing back the documents to the organization that created them or to its representatives, then destroying them). Archive retrieval can be

complete (misleadingly called a "complete retrieval") or partial (*Partial Archive Retrieval*, misleadingly called a "partial retrieval").
See Also **Archive Restitution**.

Arcsys
ERM published by Infotel. Arcsys refers to both the Arcsys Core product and all of its connectors and options.

Arcsys Connector
An Arcsys connector is an operational module generally requiring an additional license used to interface with an external software package (ECM, ERP, Mail) for archiving and/or archive retrieval to and from Arcsys.

Arcsys Core
The Arcsys Core represents all "essential" Arcsys modules, which are: Arcsys Database, the Arcsys RMI, TCP/IP and SOAP API, the Arcsys REST API, the Arcsys Transfer Server, the Arcsys Transfer Service, the Arcsys Engine, the Arcsys Web Agent, the Arcsys Application Agent, the Arcsys Auto-Archive Agent, the ArcFF format control module, the CopyRequestManager, the Arcsys standard Clients, the ArcsysFsComparator File systems comparator, the ArcProofFolder Proof Folder module and the ArcsysBatchs batch module.
See Also **Arcsys**.

Arcsys Engine
Central archiving platform on which synchronous and asynchronous archiving, indexing and retrieval processes operate. The engine can spread threads over multiple processors. This guarantees dialogue and traceability between the agents that are associated to it.

Arcsys Option
Arcsys options are added to the Arcsys Core for additional functionalities. They do not necessarily require an additional architectural module. They may be subject to a separate license. The main options are:

- ArcAFP Option (AFP format management)
- ArcMover Tape Option (media manager managing file systems and tape libraries)
- ArcIP (record ingestion)
- ArcEP (record extractor)
- ArcPAK Option (record compression on ArcMover and native ingestion of compressed files)
- ArcRFT Option (full text search)
- ArcSIGN Option (internal digital signature generation) and ArcVERIF (external digital signature verification)

- ArcCrypt Option (encryption of data at rest)
- ArcCFN (digital vault)
- ArcREF Option (record ingestion by reference)
- ArcMOVS3 Option (media manager allowing to archive and retrieve data on any Cloud media compatible with the Amazon S3 REST API)

Attestation policy

An attestation policy allows to define which type of attestation can be generated as well as a set of parameters concerning their generation.

Classification Scheme

A classification scheme in archiving and digital preservation refers to an organized framework for categorizing records and archival materials based on a hierarchical structure. It facilitates systematic retrieval, management, and preservation of information. In the context of Arcsys, the classification scheme is defined as the structural entity that contains a hierarchy of classes. These classes are used for organizing archives and records and for implementing specific archival policies such as retention schedules and format management. Within Arcsys, a classification scheme is linked to a specific repository, providing an organizational backbone for multiple collections. It also serves as a navigational tool for end users, enabling them to explore archives through the hierarchical structure of classes, alongside navigation by repository and collection.

Collection

Set of rules that a record must comply with. The collection is defined via the Web agent or Arcsys API, and comprises information contained in the relational database tables. A collection always refers to two rules: one concerning the **storage policy** and one relating to the **indexing mask**. A collection is assigned to the record on the initial record request.
See Also **Storage policy**, **Indexing mask**.

Deletion

MOREQ2010 provides the following definition for this concept: the act of deleting data from the relational database so that no trace remains. Generally speaking, an entity can only be deleted if is not used in a stored record. Otherwise, it can only be destroyed and not deleted, thus leaving a residual entity.
See Also **Destruction**.

Destruction

Irreversible action that deletes the documents by applying disposal criteria. It can be associated with the retention of residual information in the relational database.

Disposal

Outcome of archived documents when the retention period ends, i.e. generally, destruction or transfer.
See Also **Destruction**, **Transfer**.

Disposal due date (or retention end date)
Scheduled end of retention date.

Disposal Hold
Arcsys can be used to place a "disposal hold" on one or more lots archived in the application. This prevents certain sensitive operations, such as transitioning the lots to end-of-life status or migrating them to a different storage medium. All other operations remain authorized. The disposal hold guarantees that no irreversible change affecting the archival integrity or status of the lot can occur while the hold is active.

Electronic Attestation
Document produced to attest that an action or an electronic transaction has occurred.

Envelope
Arcsys groups documents stored in the system in envelopes, either created by Arcsys during the archiving process (in this case, files in TAR format), or created prior to Arcsys processing by the user or third-party processes (*native envelopes* in AFP or ZIP format, for example). The representation of an envelope in the Arcsys Database is called a **logical envelope**. Its technical implementation is also called *MoverReference*. Last but not least, the representation of information of where the envelope is physically stored in the optional ArcMover module is called *MoverMedia*.

Event
In Arcsys, a retention schedule can associate the start of record retention with an event with a known or unknown date. This event, created in an Arcsys repository, can thus be attached to a number of different retention schedules.
See Also **Retention schedule**.

Feature preview
A Preview status on a feature enables early access to non-production features, allowing users to explore and provide feedback for improvement.

Features in Preview status should not be used in production environment, as they are not fully implemented yet.

Fixity
The quality of a document that has not been subject to intentional or accidental destruction, alteration or modification.

Format policy
A format policy is used to define a set of rules concerning format checks for a given file type. These rules are used to specify the action that will be performed, the expected results of these actions, as well as the error cases, triggering archiving failure.

Hash value
Also called an "integrity certificate" in cryptography, the hash value is the digest of a message which guarantees a practically unique result that is impossible to reverse calculate. The most commonly used algorithms are MD5 (128 bits), SHA-1 (160 bits), SHA256 (256

bits) and SHA512 (512 bits). Arcsys includes a module that is capable of dynamically calling several algorithms. The choice of an algorithm type is valid for all archived objects within the same Arcsys product version; compatibility with algorithms from the previous version is guaranteed. The associated term *hash function* is also used.

Indexing mask

As is the case with the storage policy, an indexing mask is a rule that is referenced by a collection. An indexing mask can be referenced by several collections. An indexing mask refers to the use of a set of Keyword = Value pairs. The keyword component is set to make sense in a specific business application (e.g. Accounting Day, Department, Account No., Account Holder, etc.). The value component can be either unrestricted, or restricted to a set of acceptable values (e.g. A, B or C), or in date format, or restricted by an input mask. Some pairs are defined as mandatory whereas others may be optional.

An application which uses an indexing mask through a collection must supply all Keyword=Value pairs as they are defined using this mask. Any indexing-related errors lead to the record being rejected for conformity. This record is then added to the list of records with errors.

The indexing mask is defined by an administrator via the Arcsys interface or APIs. It is comprised of a set of metadata element definitions.

Journal

A journal is an XML file which contains a list of PREMIS events.

Lot

Arcsys can consolidate several different objects that form a functional set in a client application in the same physical record. It is comprised of different types of objects: files, databases, or any other type of object managed by Arcsys. It is possible to retrieve the entire lot or one of the objects contained in the lot. The MOREQ2010 record is translated in Arcsys implementation by a lot; the lot, as opposed to a MOREQ2010 record, can represent documents that are not yet archived.

Lot enrichment

The process of adding new objects to an existing archive.

Manifest

The manifest is an XML file that defines precisely the content of a record. The manifest contains: metadata associated with the record, structure metadata, a description of the physical files of the record(s) that follow, the object-by-object content of the record, object formats, object names, their size, hash value, the algorithm used to calculate the hash value, etc. The manifest is a type of complete ID card for the record.

The manifest is always written on the storage media and precedes the record that it describes. This process is used to automatically describe storage media (irrespective of the medium). With this system, users can understand media content and metadata without installing the software that generated the records.

Metadata element definition (or keyword)
Component of an indexing mask. We use the term "metadata element definition" rather than the term "keyword" as it is closer to MOREQ2010. The metadata element definition in particular defines the type of metadata (date, string, digital, controlled) and its input mask, for example.
See Also **Indexing mask**.

Object
The object is a basic archived unit that can be retrieved via Arcsys. Lots contain one or more objects. An object can be: a file, a directory, a table, a relational table, etc. The MOREQ2010 component is implemented by this object concept; the object, as opposed to a MOREQ2010 component, can represent a document that has not yet been archived.

Online
Storage level, which must be disk type, that makes records permanently available within an extremely reduced time period.

Permissions
Permissions refer to the user profiles or groups authorized to access documents or sets of documents archived in the system.

Program exit
Place in the standard workflow for picking up and executing specific code.
See Also **Workflow**.

Proof folder
A proof folder consists of a record, a proof slip, and, where appropriate, additional items (common signature or timestamp response, for example) that are used, by demonstrating the fixity and the authenticity of a document, for admission as proof. A proof slip can be generated using Arcsys Web Agent, ArcWeb Module, or Arcsys REST API. A proof folder can only be generated using ArcEP.

Record
A record is an evidential document that is deemed sufficiently important by the creator to be managed by an ERM that will manage its life cycle (retention, disposal, etc.). A record represents an archived lot. A record is archived via a *record request*. Archiving a document *creates a record*.

Relational database (or referential)
Essential component of the system, it contains all the data (excluding archived data) used by Arcsys for its operation. It includes logical entities called "repositories" (see definition).

Repository
Logical entity in the Arcsys relational database. The company can define as many repositories as it wants, either to define a test set, to isolate an application, or for any other reason. These repositories are entirely independent of each other. They all have their own pattern and all have the same structure.

Restore(or retrieval)
This term is used misleadingly in Arcsys to refer to the concept of archive retrieval. It is not accepted in archiving terminology as to mean transfer and then destruction.
See Also **Archive Retrieval**.

Retention and disposal schedule
This comprises all the rules defining the record retention period for a company or an organization, according to risks of unavailability and information system access requirements. It specifies the disposal after these time periods.
See Also **Retention schedule**.

Retention period
A duration expressed in days, months or years of object retention. The retention period is a concept used notably in MOREQ2010.

Retention schedule
A retention schedule defines the start and the end of the retention of records that are attached to it, either directly or through their class.

Retention start date
Date from which a retention period must be taken into account. The retention start date is a concept used notably in MOREQ2010.

Security
An ERMS requirement that involves including documents whose use (confidentiality, risk of exposure) and/or fixity (non modification of content, non-alteration of media) should be closely monitored.

Storage policy
A storage policy is a rule that is referenced by a collection. The policy dictates the storage media which are successively implemented to hold a record, as well as the retention period for each media. The storage policy is defined through the graphical interface. Applications or business users use it indirectly through the reference to a collection. A storage policy can be changed over time to reflect new retention periods or new storage media. The policy covers storage units by logical pool.

Storage pool
Logical storage pool, characterized in particular by its time period (e.g. 10 years). The storage policy assigns a "zone" to a "policy".

Storage zone
The storage zone is a logical entity representing a physical storage space (e.g. set of file systems, tape libraries, cloud storage).

Synchronous retrieval
Archive retrieval that takes place in the form of a direct retrieval of a document (for direct viewing or downloading) in a Web browser.

See Also **Archive Retrieval**.

Time stamping

Time stamping is a technique used to associate a document with a certain date in reference to a given and recognized time system. The date set in this way is an essential element for document authentication. Time stamping can be performed internally or by a third-party time stamp.

Tracking

Result of continuously creating, capturing and maintaining information about the movement and use of the system and objects (ISO 15489-1:2001, 3.19).

Transfer

In an archival sense, this operation sends an archived object to another IT system. Once the transfer is performed, the object can be removed from the ERMS as needed. In OAIS terminology, a transfer represents more specifically the physical transmission of a record or a set of records by a service supplying an archive service. Not to be confused with the transfer of data in the purely technical sense, as with the Arcsys Transfer Server module.

Transit Zone

Entity logically connecting an application agent and a directory, along with additional configuration.

Workflow

A set of operations carried out from the beginning to the end of a process. In Arcsys, this refers to all actions carried out on archives and objects, either directly or indirectly, in the case of archives, from their pre-archiving or preparation to their removal from the system (after they have reached end-of-life). There are other workflows in Arcsys besides the archiving workflow, which are more administration-oriented. Customized workflow involves the use of at least one drop-off point to carry out customer processing.

Registered Trademarks

Firefox is a registered trademark of the Mozilla Foundation.

Linux is a registered trademark of Linus Torvalds.

UNIX is a registered trademark of the Open Group.

Microsoft Windows, Windows NT, Windows Server, SQL Server, Internet Explorer are registered trademarks of Microsoft Corporation in the United States and/or other countries.

SAP is a registered trademark of SAP AG in Germany and other countries.

MySQL is a registered trademark of Oracle and/or its subsidiaries. MariaDB is a registered trademark of Monty Program Ab.

Java is a registered trademark of Oracle and/or its subsidiaries in the United States and other countries.

Infotel is a registered trademark of Infotel SA.

All trademarks mentioned are the property of their respective owners.

Infotel Technical Support

https://techsupport.infotel.com