Arcsys API

API Guide

26.1.STS
March 12, 2026

Version: 26.1.STS

Publication date of the document: 2026-03-12

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.

This document is part of the Arcsys software package, a product designed and developed by Infotel. All rights are reserved for Infotel.

Contact details:

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

Table of Contents

Preface

1. Introduction

This document describes the APIs of the Arcsys product.

2. Reference Documents

2.1. Concepts

Arcsys Presentation Manual: **Arcsys-presentation-26.1.0.STS-en.pdf**

Arcsys Functional Description Manual: **Arcsys-functional-description-26.1.0.STS-en.pdf**

2.2. Installing and Updating

Arcsys Prerequisites Manual: **Arcsys-requirements-26.1.0.STS-en.pdf**

Arcsys Installation Manual: **Arcsys-installation-26.1.0.STS-en.pdf**

2.3. Operations

Arcsys Administration Manual: **Arcsys-administration-26.1.0.STS-en.pdf**

Arcsys Errors Manual: **Arcsys-error-26.1.0.STS-en.pdf**

2.4. GUI

Arcsys Web Interface User Manual: **Arcsys-web-26.1.0.STS-en.pdf**

Interface Guide: **Arcsys-web-end-user-26.1.0.STS-en.pdf**

2.5. Development

Arcsys API Manual: **Arcsys-api-26.1.0.STS-en.pdf**

2.6. Option guides

ArcHP Option Guide: **Arcsys-option-archp-26.1.0.STS-en.pdf**

ArcREF Option Guide: **Arcsys-option-arcref-26.1.0.STS-en.pdf**

2.7. Optional modules

BatchReporting: **BatchReporting-UserGuide-26.1.0.STS-en.pdf**

ClassAssigner: **ClassAssigner-UserGuide-26.1.0.STS-en.pdf**

MetadataReplacement: **MetadataReplacement-UserGuide-26.1.0.STS-en.pdf**

StartRetentionDateAssigner: **StartRetentionDateAssigner-UserGuide-26.1.0.STS-en.pdf**

3. Symbols and Meanings

Note

Identifies information of particular interest

Important

Identifies important information

Part 1. Overview

1. Getting started with Arcsys APIs

1.1. Role and nature of the Arcsys APIs

There are two types of APIs in Arcsys:

- *External* APIs: these APIs can be used for external developments. They enable the access to the information found in the product relational database via programming, without prior knowledge of the internal structure of the tables used. For example, you may archive or retrieve data, get statistics on records... They provide continuity for any custom developments made for the product, making them independent of any future developments. External APIs are documented and supported. Any removal of method, service or interface is preceded by a deprecation for several major versions.

  External APIs may also be used internally in Arcsys.

- *Internal* APIs: these APIs are intended to be used only by Arcsys Core, optional modules or connectors. Their services are not documented.

2. External APIs

2.1. Introduction

The external APIs are:

- Arcsys REST API
- Optional API provided by ArcHP Option
- Deprecated external APIs which are Arcsys RMI API and Arcsys SOAP API

2.2. Arcsys REST API

It complies with the architecture recommended by REST. It is installed separately from the Arcsys RMI, TCP/IP and SOAP API. This module is described page 6, « Description of Arcsys REST API ».

2.3. Additional APIs

ArcHP Option provides additional features on top of existing APIs. It is described page 22, « Options and APIs ».

2.4. Deprecated external APIs

2.4.1. Introduction

Deprecated external APIs are documented and supported. However, since they are deprecated, no further method will be added.

Important

Implementing a new development using these APIs is strongly not recommended.

2.4.2. Arcsys RMI API

This API is included in the Arcsys RMI, TCP/IP and SOAP API. Its role is to provide a Java RMI interface to access to the Arcsys Database. It is used internally by modules such as Arcsys Auto-Archive Agent, Arcsys standard Clients, ArcsysFsComparator File systems comparator, and by optional external modules such as ArcIP (before version 6.0), ArcEP...

Important

Since it is used internally, Arcsys RMI API is still a mandatory module for Arcsys.

2.4.3. Arcsys SOAP API

This API is included in the Arcsys RMI, TCP/IP and SOAP API and is based on the SOAP protocol. Contrary to the Arcsys RMI API, it is not used internally by Arcsys

Important

Only the standard part of the Arcsys SOAP API is deprecated; the ArcHP Option in the SOAP API is not deprecated.

3. Internal APIs

3.1. Arcsys TCP/IP API

This module is included in the Arcsys RMI, TCP/IP and SOAP API. It is only used by Arcsys Transfer Server in order to access the Arcsys Database.

3.1.1. Installation and infrastructure

This module is mandatory and must be installed at least once on each Arcsys site, and may be located on any server which has granted access to the Arcsys Database and the external LDAP server. Please refer to the Arcsys Installation Manual and Arcsys Administration Manual for further details on installation and configuration.

Part 2. Description of Arcsys REST API

1. Introduction

1.1. Purpose of the API - Objectives and Location

The Arcsys REST API enables the actions to be performed via programming, such as creating lots and objects or creating and tracking archiving and retrieval requests, through the web interface and standard command lines.

These web services are designed separately from the internal APIs because they do not comply with the same architecture. In fact, they adopt the REST architecture, as they are *resource-oriented*. In addition, the connection to these web services takes place using OAuth2 authentication tokens hosting the access rights information. This allows these web services to be deployed and used in a clustering environment.

1.2. Installation and configuration

Arcsys REST API module must be installed on any server which has granted access to Arcsys Database and to the LDAP server. Please refer to the Arcsys Installation Manual and Arcsys Administration Manual for further details on installation and configuration.

2. Authentication

2.1. Overview

The authentication complies with the *RFC 6749 - The OAuth 2.0 Authorization Framework*, in particular with the *Resource Owner Password Credentials Grant* (chapter 4.3). The server acts both as an authorization server and a resource server.

It consists of exchanging credentials (user name with password) for authentication tokens used to access resources.

The server checks the credentials using the associated identity relational database (LDAP directory) and retrieves the access rights associated with the identity to write them in the tokens.

The tokens supplied comply with *RFC 7519 - JSON Web Tokens (JWT)*. They are signed in compliance with *RFC 7515 - JSON Web Signature (JWS)* using an HMAC hash associated with a secret key stored on the server. They must be used as described in the *RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage*.

2.2. Security Recommendations

In compliance with the recommendations of the RFC 6749, the client (i.e. application using the web services) must be considered as a trusted party by the *resource owner* (i.e. the user). In fact, the client will have access to the user credentials (name and password).

It is highly recommended that you access authentication services via a connection with a transport layer security (TLS) to correctly identify the server and to ensure the confidentiality of the exchanges.

2.3. Access token

The access token is used to access a resource. As long as an access token is valid, the server does not verify the identity declared in the token regarding the identity relational database. Only the token signature is checked; this is why it has a short validity period by default. There is an automatic refresh token mechanism that must be taken into account. This is described more precisely in the next sections.

2.4. Obtaining Tokens

Tokens are obtained by manipulating the `/api/oauth/token` resource in compliance with the *Resource Owner Password Credentials Grant*.

The access to the `/api/oauth/token` resource requires the client to be authenticated using the `Authentication` header. This header must contain the client ID (as configured in *`CLIENT_DETAILS_CLIENT_ID`* parameter) encoded in base64 and prefixed by `Basic`.

For example, a request for tokens for the user `johndoe` and the password `A3ddj3w` results in the following request:

```
POST /api/oauth/token HTTP/1.1
Host: server.example.com
Authorization: Basic Y2xpZW50X2lk
Content-Type: application/x-www-form-urlencoded
grant_type=password&username=johndoe&password=A3ddj3w
```

`Authentication header` (`Authorization: Basic Y2xpZW50X2lk`) corresponds to the client ID encoded in base64 (in this example, the chosen client ID is `client_id` which is `Y2xpZW50X2lk` in base64).

If the client authentication is successful and if the user credentials are correct, the server response will be similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "access_token":"eyJh[...]VCJ9.eyJh[...]AifQ.DRnp[...]Hj_8",
    "token_type":"bearer",
    "expires_in":300,
    "scope":"all",
    "jti":"b4ec6cec-fa71-495c-aa29-22c8c3c1be20"
}
```

To improve readability, certain parts of the tokens have been replaced by *[...]*.

Another example with a curl request:

```
$ curl -H "Authorization: Basic Y2xpZW50X2lk" http://server.example.com:8090/api/oauth/tok
en --data "grant_type=password&username=johndoe&password=A3ddj3w
```

2.5. Using an Access Token

To be able to access protected resources, a valid access token (obtained as described previously) must be sent. The transmission of the token takes place in the HTTP header `Authorization` by using the `Bearer` schema.

For example, calling the `status` resource will result in the following request:

```
GET /api/status HTTP/1.1
Host: server.example.com
Authorization: Bearer eyJh[...]VCJ9.eyJh[...]AifQ.CPYW[...]xYck
```

Important

Tokens are considered sensitive data since they carry the user credentials. This is why we strongly suggest you to use connections with transport layer security (TLS) for all calls where these tokens are used, i.e. for all calls of a resource exposed by the Arcsys REST Web Services.

2.6. Access Token refresh mechanism

For security reasons the token obtained by the client is valid only for a short time by default, i.e 5 minutes (*ACCESS_TOKEN_VALIDITY* is set by default to 300 seconds), as suggested by the RFC RFC6749.

5 seconds before the token expires, or immediately after expiration, the server includes a new access token in the Authorization response header.

This new access token must be used in the next request sent to the server.

The parameter *TOKEN_IDLE_TIME* enables you to define an idle time (starting from the moment the access token expires) during which the token can still be refreshed. If this idle time is exceeded, the user must re-authenticate himself by initiating a new authentication flow.

Example with *TOKEN_IDLE_TIME* set to 1800 seconds (30 minutes):

- A user sends a first REST request with the first access token.
- 8 minutes later, the user sends a second REST request. The request is accepted, and a new access token is returned in the `Authorization` response header.
- 2 minutes later, the user sends a third REST request:
  - If the user does not use the new token and still uses the original one, the request is rejected with a **401 Unauthorized error**.
  - If the user uses the new access token, the request is accepted.
- If this user waits 36 minutes without sending any request, and then sends a fourth REST request, it will be rejected with a **401 Unauthorized error** due to exceeding the idle time. Re-authentication is required.

Important

When developing a program connecting to Arcsys REST API, the developer must carefully manage the token, especially when multithreading is involved (when the new token is given in the

"Authorization" header, it will not be given a second time when another REST request with the "old" token is received). To simplify this mechanism, each thread can also have its own token and refresh mechanism.

2.7. Caller identification

In addition to authentication, the REST API is able to specify the identity of its caller if this one sends a request with the header `Arcsys-Client-Id` specifying this identity. This information is traced in the logs of the component and is not used for authentication.

Note

This is useful, in particular, in the context of many client applications calling simultaneously the same operations, in order to trace back each call to its source application, for example for debugging purposes.

Important

GDPR regulations must be taken into account when defining the value of the `Arcsys-Client-Id` field ; an abstract value could be preferred, for example, to the name of a physical person.

For example, the request:

```
GET /api/status HTTP/1.1
Host: server.example.com
Authorization: Bearer eyJh[...]VCJ9.eyJh[...]AifQ.CPYW[...]xYck
Arcsys-Client-Id: TestClient
```

Produces the following logs in INFO mode:

```
Received: [TestClient] GET /api/status
```

2.8. Using SSO authentication

SSO is a single sign-on technology that allows users to log into multiple applications with a single login. Once logged into the SSO service, the user can then access multiple applications without having to authenticate to each one.

The Arcsys REST API supports user authentication using the SAML protocol.

2.8.1. SAML terminology

Security Assertion Markup Language (SAML) is an XML-based standard that allows to exchange authentication data between one service and another.

The SAML terms:

- *Service Provider (SP):* The service that provides a service to the user.
- *Identity Provider (IdP):* The system that manages the identity information of the users (username, password, ...).
- *Entity ID:* A unique ID that allows the Service Provider and Identity Provider to identify each other.
- *HTTP POST:* One of the binding options supported by the SAML protocol. HTTP POST sends the message content as a POST parameter, in the payload.
- *HTTP Redirect:* One of the binding options supported by the SAML protocol. When using the HTTP Redirect, the Service Provider redirects the user to the Identity Provider to log in and then redirects the user to the Service Provider.

2.8.2. Configuring SAML single-sign-on on Arcsys REST API

Arcsys REST API is operating as a Service Provider. The other component that is needed to enable SAML is the Identity Provider. Only Identity Providers that support SAML 2.0 Web Browser SSO can be used. For example, you can use Keycloak, an open source Identity and Access Management framework.

This guide assumes that you have an existing Identity Provider and that you already created a client profile. In your Identity Provider configuration, you have to bind the Arcsys REST API and the Identity Provider for the single logout service. For security reasons, some Identity Providers need to know authentication and logout endpoints. Below are the endpoints used by Arcsys REST API:

- Authentication endpoint: `<Arcsys REST API domain>/sso/saml/2.0/authenticate/callback`
- Logout endpoint: `<Arcsys REST API domain>/sso/saml/2.0/logout/callback`. The Identity Provider has to call this endpoint to disconnect the user from the Arcsys REST API.

To configure the Arcsys REST API, you need information that can be found in the SAML metadata document, that defines the capabilities and features of your Identity Provider. The SAML metadata document can be downloaded within the Identity Provider administration interface.

Note

The Arcsys REST API cannot use the SAML metadata document directly. You need to fill in the Arcsys REST API configuration file. For more details, please refer to the Arcsys Administration Manual.

2.8.3. How to use it?

Your custom web application would need to perform the following steps in order to authenticate a user with SAML against Arcsys REST API:

1. Make an HTTP POST request to `/sso/saml/2.0/prepare`. In the request body, you need to specify the `clientId` (only used for logs) and the URL to redirect the user after authentication.

2. Handle the response from `/sso/saml/2.0/prepare`. This response contains the SAML redirect request. Your custom web application must redirect the user's browser to the URL that was returned in the response.

3. Handle the response after the user is successfully authenticated. The Identity Provider redirects to Arcsys REST API with the SAML response. Arcsys consumes this SAML response and generates an OAUTH Access Token. Arcsys redirects the user to the redirect URL set in the first step. A query parameter `accessToken` is added to it.

4. You can call operations in Arcsys REST API with the access token.

Your custom web application can log out the user by following these steps:

1. Make an HTTP POST request to `/sso/saml/2.0/logout`. In the request body, you need to set the redirect parameter to redirect the user after the log out.

2. Handle the response from `/sso/saml/2.0/logout`. This response contains the SAML redirect request. Your custom web application must redirect the user's browser to the URL that was returned in the response.

3. The Identity Provider logs out the user and redirects to Arcsys. Arcsys invalidates the user token and redirects the user's browser to the redirect URL set in the first step.

3. Use

3.1. Principles

The Arcsys REST API does not have a session or transaction mechanism. Each operation is automatic and self-checked.

Authorization checks (permissions) and data partitioning take place using information obtained from the authentication token used on each request and obtained as described in the previous chapter.

3.2. Catalog

A REST web service catalog is available. This interface based on *Swagger* is used to display the list of REST resources available with the actions available for these resources. To access it, go to the following address in an Internet browser: https://server.example.com:8090/apidocs [1] by replacing *server.example.com* with the server address where the web services are deployed.

Important

The access to the different resources must be done by specifying the Arcsys REST API version in the endpoint (except for OAuth2 authentication where no version must be specified). For example, to retrieve the Arcsys Application Agent with identifier `1` in the first version of Arcsys REST API, the following endpoint should be used: https://server.example.com:8090/api/v1/agents/1. If there is no version specified, the version used will always be the last one available, which can cause incompatibility problems depending on the evolutions of the module.

The home page is available without authentication, however, an access token is required to browse the catalog. The token is obtained in line with that described in the previous chapter. An example of a *cURL* command is displayed in the home page to simplify the procedure for obtaining a token.

3.3. Compatibility rules

It is highly recommended, when your development is starting, to use the latest available version of the API, and to try upgrading your developments to a new version when it is released. There is no programmed deadline for the support of a given version of the API (for example v1). The end of support of an API version will be announced at least two years ahead of time.

[1] https://server.example.com:8090/apidocs

The latest version of the API remains the same during the life cycle of a LTS of the product. A new LTS will generally bring additional REST operations that will be added to a new version of the API, or introduce breaking changes that will also be in a new version.

Important

The current API version is `v4` and will remain till the Arcsys 2027.1.LTS.

The API versions `v1` and `v2` are deprecated from 2024.1.LTS and will be removed from 2027.1.LTS.

Any client development using an API operation in a given version (for example, `v1`) is guaranteed to continue working when upgrading Arcsys, with exceptions (e.g. a change is needed in Arcsys model).

Note

A client development must ignore additional unknown properties in the JSON to continue working when upgrading Arcsys.

For example, in Java language, ignoring additional unknown properties can be done with the following annotation:

```
@JsonIgnoreProperties(ignoreUnknown=true)
```

Note

On the latest API version, new operations may be added and compatibility might be broken between several STS of the same LTS version. If you develop by using operations introduced in a STS version, there is no warranty another STS version of the product will not break the compatibility of the development.

3.4. Return codes

Each call to an operation of the REST API results in a return code that makes possible to deduce whether the operation was successful, and if not, what type of error it is. You can find the main return codes below.

Value	Meaning
200	The operation was carried out successfully.
204	The operation was carried out successfully and there is no content in return.
400	The HTTP request is not properly formed (possibly a missing or inappropriate parameter).
401	Authentication information is missing to perform the operation (OAuth2 token not provided or expired).
403	The operation is not allowed.
404	An entity could not be found via its identifier in Arcsys during the process.
500	An internal server error occurred. This error can be caused by a technical error (such as a lack of database access) or a violation of a business rule (for instance, using a code that already exists when creating an entity). The response body is a JSON entity that contains the following fields: "exception" (the Java class of the exception), "message" (the message associated with the exception), and "cause" (the Java class of the underlying exception).

Important

The content of the "exception", "message" and "cause" fields for a 500 error may change across different versions for the same error. Therefore, you should not rely on the content of these fields for your development purposes.

4. Advanced Operations

4.1. Archiving APIs

All operations are available in the `/upload-management` resource.

You can create a whole archive (lot, object, metadata…) with a single call. It is also possible to upload the documents to be archived through the API. A checksum is computed to check the integrity of the file during its transfer.

To create an archive and upload files in an archive:

1. You need to ask for creation of a *workspace* (this is a folder that will be created inside the *`WORKSPACE_DIRECTORY`* folder) to upload the file into the API, with the operation `GET /upload-management`. A *workspace* is a place on the system where you can upload temporary files to bind them later in the archive creation.

2. You can then upload files in this workspace using `POST /upload-management/{workspaceId}/files/{fileId}`.

   The `fileId` is a unique identifier (inside the workspace) you have to assign to the files you upload.

   The way to upload the files must be compliant to the **multipart/form-data** specification (https://tools.ietf.org/html/rfc7578).

   Each upload operation returns:

   - a "transfer" checksum (in MD5 algorithm) so that you can check that the transfer operation completed successfully.
   - optionally, if you specify "SHA-512" for the `persistentChecksum` field, a SHA-512 checksum. This checksum is called "persistent" in opposition to the "transfer" checksum because it uses a stronger computation algorithm which makes it eligible for the checksum of the documents stored in Arcsys. You can later set this checksum in the `hashValue` field of the object.

3. Finally, you can create and archive the lot with `POST /upload-management/_archive`. The objects in the lot are linked to the files you have uploaded through the `fileId` attribute.

   The archive will be processed by one of the application agents configured in the parameter *`AGT_CODES`* (see in Arcsys Administration Manual) of the API.

4. The uploaded files are automatically cleaned by default. See the parameters *`WORKSPACE_CLEAN_EXPIRED_WORKSPACE_FILES`*, *`EXPIRED_WORKSPACE_FILES_CLEANE`*

R_INTERVAL, *WORKSPACE_FILES_LIFESPAN* in Arcsys Administration Manual for further details on the configuration of the cleaning process.

Important

If the APIs are used in cluster with ArcHP Option, the *WORKSPACE_DIRECTORY* directory must be shared between the different APIs (for example with a NFS shared folder).

4.1.1. Preparing an archive: saving an archive for later submission

It is also possible to proceed to the creation of the archive without archiving it immediately. This allows to **prepare an archive** in several steps. For example, this allows to create the archive first, add some documents in it, fill in some metadata, then add other documents and submit additional metadata, before submitting the archival operation.

After having uploaded files in a workspace with `POST /upload-management/{workspaceId}/files/{fileId}`, the creation of the archive without creating an associated archiving request must be done with `POST /upload-management/_prepare_archive` instead of `POST /upload-management/_archive`.

Adding documents in a prepared archive is then possible with `POST /upload-management/lots/{lotId}/objects`. This operation can be called on any archive whose lifecycle is not "archived" or "end of life", and whether it has been created through a REST operation or not.

When the archive is ready, the archive operation can then be triggered on one or several archives at once with `POST /upload-management/_archive_prepared_archive`.

4.1.2. Adding uploaded objects to an archived lot

These are the operations enabling archive enrichment as described in the paragraph `Adding an object` in Arcsys Functional Description Manual

After uploading a file in a workspace with `POST /upload-management/{workspaceId}/files/{fileId}`, it is possible to add the document with `POST /upload-management/lots/{lotId}/objects/_addObject`. This operation can only be called on lot which life cycle is "archived".

When the archive is ready to be enriched, the enrichment operation can then be triggered on one or several archives at once with `POST /upload-management/lots/requests/_enrich`.

4.1.3. Checking if a document is already archived

The `POST /upload-management/_objectsPresence` operation returns a list of documents which checksums match the uploaded file ones.

This operation calculates the hash of the provided file using one or more hash functions, which can include MD5, SHA1, SHA256, and SHA512. If the `hashFunctions` parameter is empty, the hash is computed with all available hash functions.

All objects that match any of the computed hash values are returned, provided that the user has permission to view them. The objects may belong to any archive or collection, and the archive can be in any state (e.g., archived, end of life, error).

4.2. Retrieval and unique identifiers of documents

All operations are available in the `/download-management` resource.

You can download a document from its database identifier (the pair archive id, object id) or from its *unique identifier* (as described in the next section).

Used together with the ArcHP Option, a cluster of REST API may be used to provide high performance for multiple parallel downloads.

4.2.1. Generating an unique identifier for a document

You can obtain the *unique identifier* (abbreviated "uid") of a document with the operation `/download-management/lots/{lotId}/objects/{objectId}/uid`.

This unique identifier follows the URI scheme given by the ARK (Archival Resource Key) standard : `ark:/NAAN/UUID`. The NAAN (Name Assigning Authority Numbers) has 5 digits and is currently hardcoded to the value **11630** (it's Infotel's NAAN). The UUID is a type 4 UUID (unique identifier on 32 hexadecimal characters).

Here is an example of a unique identifier: `ark:/11630/61D780F954C94224BC8E98E1C282AAFB`

4.3. Saving an archive search

All operations are available in the `/search-management` resource.

An archive search (only the criteria, not the results) can be saved to, read from, or deleted from Arcsys database.

It can be private (associated to a given user) or public.

4.4. Generating a proof slip

Operations to generate proof slips are available in the `/download-management` resource.

It is possible to generate a proof slip:

- At document level via the operation `/lots/{lotId}/objects/{objectId}/proof_slip`;
- At archive level via the operation `/lots/{lotId}/proof_slip`.

For a proof slip at lot level, an additional section lists various information about each document of the lot such as identifier, path, fingerprint, metadata and signature information.

The lot concerned by these two operations may be in end of life.

As explained in Arcsys Functional Description Manual, the role **PROOF_FOLDER** is required to be able to generate a proof slip.

4.4.1. Customization

Some characteristics of the proof slip can be customized by modifying the `etc/extractpreferences.xml` configuration file.

You can modify:

- The language of the proof slip, by changing the value of the `LANG` property to `EN` or `FR` (default) (example: `<PROPERTY NAME="lang">EN</PROPERTY>`);
- The type of hash value for the proof slip (as described in the "Authenticity of the proof slip" section of the Arcsys Functional Description Manual), by changing the value of the `HASH_TYPE` property to `SHA1`(default), `SHA256` or `SHA512`;
- The inclusion in the proof slip of the various request types, and the maximum number of request displayed for each type, with the `INC_ARCH_xxx` and `NB_ARCH_xxx` properties.

4.5. Downloading a list of archives as a zip file

You can download a list of archives as a zip file with the operation `/download-management/_zip`.

For each archive in the list, it is possible to specify only a set of documents to download instead of downloading the whole archive.

Important

There is no control over the total number of items to download or the maximum size of the zip file. This control must be implemented on the client side.

Part 3. Options and APIs

1. ArcHP Option APIs

1.1. Purpose of ArcHP APIs

ArcHP Option allows calling methods or operations in the context of a **cluster of APIs**, but not on all available APIs:

- The use of ArcHP Option is **not compatible** with RMI APIs.
- The use of ArcHP Option is compatible with a **specific set of dedicated stateless** SOAP APIs.
- The use of ArcHP Option is compatible with **all operations** of the Arcsys REST API, providing some conditions in specific cases.

1.2. ArcHP Option SOAP Methods

1.2.1. Introduction

The deprecated Arcsys RMI, TCP/IP and SOAP API provided methods based on a "stateful" model, as authentication must be performed initially by the `connect` method and the authentication token is sent to the methods called subsequently.

This operating mode presents issues regarding load distribution when implementing a HTTP load balancer between the client and various Arcsys RMI, TCP/IP and SOAP API instances.

The ArcHP API provides a set of "stateless" methods compatible with the use of a load balancer. Each call of those methods requires authentication.

The ArcHP API consists of the three following methods:

- `statelessSearchArchives` - to perform a search
- `statelessSearchArchivesCount` - to get the number of results returned by a search
- `statelessOnlineRestore` - to perform a synchronous document retrieval

1.2.2. Access to the Methods

The *`WS_API_ARCHP_SERVICE_NAME`* parameter defines the service name of the ARCHP API. `ArcHPWS` is the default service name.

The *`WS_API_PORT_NUMBER`* parameter in the `ARCSYS_WS_API.properties` file sets the port number. Note that this is the same port as the internal SOAP API.

1.2.3. `statelessSearchArchives` and `statelessSearchArchivesCount` methods

The `statelessSearchArchives` method performs record searches by sending the lots or objects corresponding to the search criteria.

The `statelessSearchArchivesCount` method does the same but returns the number of results instead of the actual results.

1.2.3.1. Parameters

- `LotSearchParameters` search parameter: This object contains the criteria (keywords, record state, etc.) corresponding to the search. For more information, see the javadoc of the `LotSearchParameters` class.
- In the case of `statelessSearchArchives`, optional parameter (`long` type) `startIndexLine`: The index of the first element to send; this allows you to iterate by range on a large number of results, for example. By default (set to `null` not specified), the list sent starts from the beginning.
- In the case of `statelessSearchArchives`, optional parameter (`integer` type) `nbLinesToLoad`: The number of elements to send; this allows you to iterate by range on a large number of results, for example. By default (set to `null` or not specified), the list sends all the elements.

Important

If the search criteria are not restrictive enough and if a maximum number is not specified, an `OutOfMemory` error can be generated in the API.

- `String` type user name: application account user name to make the call.
- `String` type password: unencrypted password of the application account to make the call.

1.2.3.2. Restrictions

To search in all the repositories, the lot-level criteria list must be empty.

To search in all the collections of a single repository, the lot-level keyword levels must be populated.

The boolean operators only work for the criteria on the same keyword.

1.2.4. `statelessOnlineRestore` Method

This method is used to retrieve an Arcsys object synchronously (called synchronous retrieval).

1.2.4.1. Parameters

- `long` type lot identifier: identifier of the lot in which the object to retrieve is found.
- `long` type object identifier: identifier of the object to retrieve.
- `boolean` Validation of the hash of the envelopes: to specify whether you want to validate the hash of envelopes (set to `false` to improve performance).
- `String` type user name: user name of the application account to be used to make the call.
- `String` type password: unencrypted password of the application account to be used to make the call.

1.2.4.2. Samples of Requests and SOAP Responses

Here is an example of the SOAP exchanges used to perform a stateless synchronous retrieva, with the following SOAP message:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="101" xmlns:ns2="http://archp.wsapi.infotel.com/">
<SOAP-ENV:Body>
  <ns2:statelessOnlineRestore xsi:type="ns1:stdClass">
    <lotId>106</lotId>
    <objectId>10704</objectId>
    <checkTarIntegrity>false</checkTarIntegrity>
    <user>user</user>
    <password>password0</password>
  </ns2:statelessOnlineRestore>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Using the lot identifier and the object identifier, you can modify this XML request and save this XML in a file; here it is called `request.xml`. In this example, the server name `example.com` and the default port and service name are used. The following command can then be run:

```
curl --header "Content-Type: text/xml;charset=UTF-8" --header "SOAPAction:" --data
 @request.xml http://example.com:50060/ArcHPWS?wsdl
```

The following response is then received:

```
--uuid:0af02e25-4efe-41c5-ae28-17e69b78b7a6

Content-Type: application/xop+xml; charset=UTF-8; type="text/xml"
```

```
Content-Transfer-Encoding: binary

Content-ID: <root.message@cxf.apache.org>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ns5:statelessOnlineRestoreResponse xmlns:internal="http://internal.components.infotel.com/
" xmlns:search="http://search.components.infotel.com/" xmlns:tools="http://tools.common.
aro.infotel.com/" xmlns:ns5="http://archp.wsapi.infotel.com/">
<file><xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
 href="cid:c0cf8730-366b-4026-8ae1-e08015845b08-1985@cxf.apache.org"/></file>
</ns5:statelessOnlineRestoreResponse>
</soap:Body>
</soap:Envelope>

--uuid:0af02e25-4efe-41c5-ae28-17e69b78b7a6

Content-Type: application/pdf

Content-Transfer-Encoding: binary

Content-ID: <c0cf8730-366b-4026-8ae1-e08015845b08-1985@cxf.apache.org>

Content-Disposition: attachment;name="/opt/infotel/ArchivingProduct/ArcsysBuffer/depotIP/
servicetest/VERS/droptest/lot1463754759650168720313876 97/DEPOT/fichier95.pdf"
[Binary content of the file]

--uuid:0af02e25-4efe-41c5-ae28-17e69b78b7a6–
```

1.3. ArcHP Option in the Arcsys REST API

1.3.1. Providing clustering thanks to the persistence of the refresh token

With ArcHP Option, you can use a cluster of Arcsys REST APIs for your client developments.

This is possible thanks to the persistence of the refresh token mechanism.

The persistence of the refresh token is a mechanism done by storing it in the Arcsys database. This allows to eventually set up a REST APIs cluster. The REST APIs will then be able to share the same refresh token.

To set up a cluster of REST APIs, it is necessary to fill the parameter *JWT_JWS_KEY_FILE* in the `arcsys-rest.properties` configuration files of all the Arcsys REST APIs that need to be used in cluster mode. This parameter must indicate the path towards a secret key that can be generated with the script `generate_key.sh` delivered in Arcsys Engine (please refer to Arcsys Installation Manual to see how to generate secret keys).

Important

All the REST API that are used in cluster mode must share the same secret key.

In order to avoid unnecessary storage in the database, a cleaning mechanism for expired refresh tokens can be activated via the parameter *`CLEAN_EXPIRED_TOKENS`* that should be set to `true` for each REST API you use.

The settings about the persistence and the cleaning of expired tokens are detailed in Arcsys Administration Manual.

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 keywords.
See Also **Keyword**.

Journal

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

Keyword

Component of an indexing mask. The keyword in particular defines the type of metadata (date, string, numerical, controlled) and its input mask, for example.
See Also **Indexing mask**.

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.

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 used to segment and isolate information within Arcsys. Each repository has its own storage profiles, indexing masks, keywords, and configuration, all fully independent from those of other repositories.

A repository also defines its own permission scope.
See Also **Storage policy**, **Indexing mask**, **Keyword**, **Permissions**.

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.

Slug
A tenant slug is a short, URL-friendly identifier used to represent a tenant in a stable and readable way within application URLs. It uniquely identifies the tenant using only lowercase letters, digits, and hyphens.
See Also **Tenant**.

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**.

Tenant
A tenant is a distinct logical environment within a shared Arcsys installation, fully isolated in terms of customization while running on a common technical platform. It enables each client or organization to benefit from a fully personalized interface and functional setup without impacting others.

Tenants do not isolate or partition the data stored in Arcsys.
See Also **Slug**.

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