# Seclore Server SDK Java

Version 4.5.2.0

# Contents

# 1. Introduction

## 1.1. About Seclore

Seclore enables enterprises to mitigate the risk arising out of information breach by allowing users to retain control in the usage of their documents even after the documents are shared with others within or outside the enterprise. This helps to enhance collaboration while at the same time reduce worries about information leakage or breaches. More details on the product can be found on the website www.seclore.com

## 1.2. Purpose of the document

This document is an extension to Seclore Web Services Interface.pdf, which documents the details of Web Service APIs exposed by Seclore Policy Server. The purpose of this document is to make the reader aware of the Seclore Helper APIs exposed using Web Services provided by Seclore.

## 1.3. Scope of the document

This document only talks about the APIs exposed as part of Seclore Helper Library. It does not explain any other concepts related to Policy Server like Credentials, Hot Folders etc. It is assumed that reader of this document is aware of the basic concepts related to Seclore. For this, the reader may refer the user manual of Seclore.

# 2. Pre-requisites

## 2.1. File IO Permission

To read and write file during operations like protect, unprotect etc.

## 2.2. Security Permission

To connect to Policy Server with non-trusted SSL certificates like self-signed SSL certificates. (To set ServicePointManager.ServerCertificateValidationCallback property and set the SSL SCHANNELS such as TLS1.2/TLS1.1/TLS1.0/SSLv3).

# 3. Seclore Helper Library

Seclore Helper Library is built on top of the RESTful web services provided by Seclore Policy Server.

Seclore Helper Library is created for applications to be able to integrate easily with Seclore. It provides easy interface primarily to protect or unprotect files. It can also be used for calling other web services exposed by Seclore Policy Server.

Before calling any web services, Policy Server expects the user to be authenticated. Once authenticated, the user can make any number of calls if the same session is continued.

User authentication can be performed either as an end user or as a HotFolder Cabinet user. These details are specified during the initialization of this Library.

Seclore Helper Library internally maintains a pool of such authenticated sessions. The caller can fetch a session from this pool when required and return it back to the pool when done. The library internally takes care of creating new authenticated sessions as and when required. One can specify the maximum number of such concurrent sessions that can be created. The Library also takes care of creating new session, if any of the

sessions has expired. It also works with multiple Policy Server URLs. If the primary (first) URL in the list is not accessible, it automatically connects to the next one and so on.

Normally, for protecting and un-protecting files, the caller is expected to just call protect and unprotect APIs. These APIs internally take care of getting an authenticated session from the pool and performing the operation on the file.

The Library is available as a Java jar file which needs to be integrated with the application that expects to use the library to protect/unprotect files.

The Library has certain API functions that are exposed to the application. The application first must initialize the library to use the functionality provided by the Library.

This Library, apart from the API functions, also provides a logging mechanism using log4j library. It exposes certain APIs to log different level of logging like error, info and debug. The Library internally also uses the same APIs to log information. In case of debug level, it logs all the steps involved in processing an API. In case of info, it logs the milestone steps. In case of error, it logs the unexpected errors.

One-line description of each of the APIs exposed within Seclore Helper Library is available below. For method signature and parameter details, please refer to the javadocs. There is a sample code available to protect/unprotect files based on this Library.

# 4. Commonly Used APIs

## 4.1. Class: FSHelperLibrary

### 4.1.1. Initialize Library

initialize () – API to initialize the Library. This API should be called before calling any other method of the Library; else all subsequent calls to the Library API will fail. This API should be called at-least once before using any APIs of the library. As an input, this takes an XML (Seclore Server SDK Java Logger Configuration Structure.xml). Details of the XML are provided along with the Documentation in the **Doc/Config** folder in the package. The application can pass custom implementation of ISecloreSDKLogger in case it wants to log in the application's logger.

### 4.1.2. Terminate Library

terminate () – API to clean up the Library. User should call this method when Application does not want to use this Library any more. This API should be called only once when the application no longer needs to call any methods on the library.

### 4.1.3. Create new FSHelper Object

initializeHelper () – API to create a new FSHelper Object. This API should be called before calling any of the methods of the library which require a Policy Server Session. This API serves as the root for calling the other methods which deal with File Operations.

### 4.1.4. Get FSHelper Object

getHelper () – API to fetch the FSHelper object. The input is the unique identifier which was provided during the initializeHelper method.

### 4.1.5. Terminate FSHelper Object

terminateHelper () – API to terminate the FSHelper object. The input is the unique identifier which was provided during the initializeHelper method.

## 4.2. Class: FSHelper

APIs defined need to be called using the FSHelper Object.

### 4.2.1. Check file is protected

isProtectedFile() – API to check if file is Seclore protected or not. This takes file path as input. Another version of this API accepts file bytes as input.

### 4.2.2. Check file is supported

isSupportedFile() – API to check if file is from Seclore supported extension list or not.

### 4.2.3. Protect file with Advanced Protection

protectX() – API to protect file with Advanced Protection, using the protection details passed. This API call should be preceded by IsSupportedFile() API call, to check whether the file format is supported for Advance Protection.

### 4.2.4. Unprotect file with Advanced Protection

unprotectX () – API to unprotect a Seclore protected file, protected with Advanced Protection only.

### 4.2.5. Check file is supported for Universal Protection

isBasicProtectionSupported () – API to check if file is supported for Basic Protection or not.

### 4.2.6. Protect file with Universal Protection

protectFileWithBasicProtection () – This API has been deprecated. Use protectAndWrap() instead.

protectAndWrap() - supports both advance and universal protection, thus this API call should be preceded by isBasicProtectionSupported() API to check if file is supported for Universal Protection or not.

### 4.2.7. Unprotect file with Universal Protection

unprotectFileWithBasicProtection () – This API has been deprecated. Use unprotectAndUnwrap() instead. Use unprotectFileWithBasicProtection() API only if there are protected files with extension '.seclore' which needs to be unprotected.

### 4.2.8. Protect and wrap file

protectAndWrap () – API to protect and create HTML wrapped file using the protection details passed. The file will be protected using either Advance Protection or Universal Protection based on the file format.

API will replace the original unprotected file with protected HTML wrapped file.

### 4.2.9. Unwrap and unprotect file

unwrapAndUnprotect () – API to unprotect a file which is a protected and HTML wrapped file. The original protected html wrapped file will be replaced with unprotected unwrapped file.

### 4.2.10. Send request

sendRequest () – Generic API to send request to Policy Server by calling one the Web Services. It uses a session from the pool, if not passed explicitly. It accepts a XML request and returns a XML response. Please refer to the Seclore Web Services documentation for more details on the XML structure of the request and response.

### 4.2.11. Wrap file

wrap () - API to create HTML wrapped file from a protected file.

### 4.2.12. Unwrap file

unwrap () - API to create unwrapped file which is a protected file from an HTML wrapped file.

### 4.2.13. Check file is wrapped

isHTMLWrapped () – API to verify whether the file is HTML wrapped or not.

#### 4.2.14. Check file is supported for HTML wrapping

isHTMLWrapSupported () – API to verify whether the file is supported for HTML wrapping or not.

#### 4.2.15. Get Seclore protected file identifier

getFileId () - API to fetch the file identifier of a Seclore protected file.

#### 4.2.16. Get Seclore HTML wrapped file identifier

getHTMLWrappedFileId () - API to fetch the file identifier of a Seclore HTML wrapped file only.

#### 4.2.17. Protect email body

protectMailBody() - API to protect the email body using protection details which have been passed.

#### 4.2.18. Unprotect email body

unprotectMailBody() - API to unprotect the protected SMail file. This method returns the unprotected content of SMail in "html" file.

#### 4.2.19. Protect and get SMail

protectAndGetSMail() - API to protect the email body using protection details which have been passed.

### 4.3. Additional Methods (FSHelper)

#### 4.3.1. Create Session

createPSConnection () – API to create a new authenticated session with Policy Server which is not part of the connection pool. Session needs to be released after use.

#### 4.3.2. Get Session

getPSConnection () – API to get authenticated session with Policy Server from connection pool. Session needs to be released after use.

#### 4.3.3. Release Session

releasePSConnection () – API to release Policy Server session back to the pool (if fetched from pool) or terminate it (if not fetched from the pool).

#### 4.3.4. Fetch protected file information

getFileHeader () - API to fetch the FileHeader XML. This XML contains the Policy Server Identifier, File Identifier and Policy Server URL's. This APIS accepts file path.

### 4.4. Logging API(s)

These are a set of APIs available in the Library, which aid the Library user to log relevant information at different levels, from the Application.

As the name suggests these APIs are used to log messages of the nature info, error or debug. Based on the level passed as parameter during initialization of library, actual logging is done. API names are logInfo(), logError() and logDebug().

## 5. Error Codes

The Error Codes are provided in a separate document. For more details on Error codes, please refer Doc/Error Codes folder in the package. Error Codes of the Server are available separately along with the Server documentation.

# 6. FAQs

## 6.1. How to use customised buffer files with FSHelper session?

**Note:** The fresh deployment of Seclore 3.18.0.0 and above does not require custom buffer files. To use default buffer files instead of custom, pass null value for pResourcePath parameter while initializing the FSHelper instance as given below.

**For custom buffer files:**
While creating the FSHelper object using the initializeHelper method of FSHelperLibrary, the resourcePath is supplied as a parameter. The ResourcePath is the directory containing the Templates directory (directory containing the customized buffer files) and the SupportedExtInfo.xml. For customized buffer files, you can supply the required directory path as the pResourcePath parameter.

## 6.2. Can I create different FSHelper sessions for the same Policy Server URL?

Yes, you can create different FSHelper sessions for the same Policy Server URL. Examples of different sessions can be a Non-Elevated EA Session, Elevated EA Session and an End User session, all pointing to the same Policy Server.

## 6.3. What is the recommended way to Terminate FSHelper object?

The recommended way to terminate the FSHelper object is as follows:

1. Dispose/Clear all references of the FSHelper object used in the calling application.
2. Call the terminateHelper method of FSHelperLibrary class.

## 6.4. How can I use the Protect, Wrap, Unwrap, Unprotect etc methods?

The above-mentioned methods can be accessed through the FSHelper object.