Seclore Online Integration - Developer Guide

Version : 3.1

Legal Notice

The software described in this guide is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

No part of this document may be copied, modified, reproduced, or distributed in any form, without the express written consent of Seclore Technology Private Limited.

Technical Support

For any technical query or assistance please visit http://support.seclore.com or write to support@seclore.com

For the latest news of upgrades, documentation, and related products, visit the Seclore web site at www.seclore.com.

Contents

1. Introducing the Seclore Online Integration

1.1. About This Guide

This guide defines the detailed technical approach and steps along with design considerations for integrating Enterprise Applications with Seclore Online. This guide is meant for technical architects and development teams who intend to integrate enterprise applications with Seclore Online.

Readers are expected to be well versed with the following aspects:

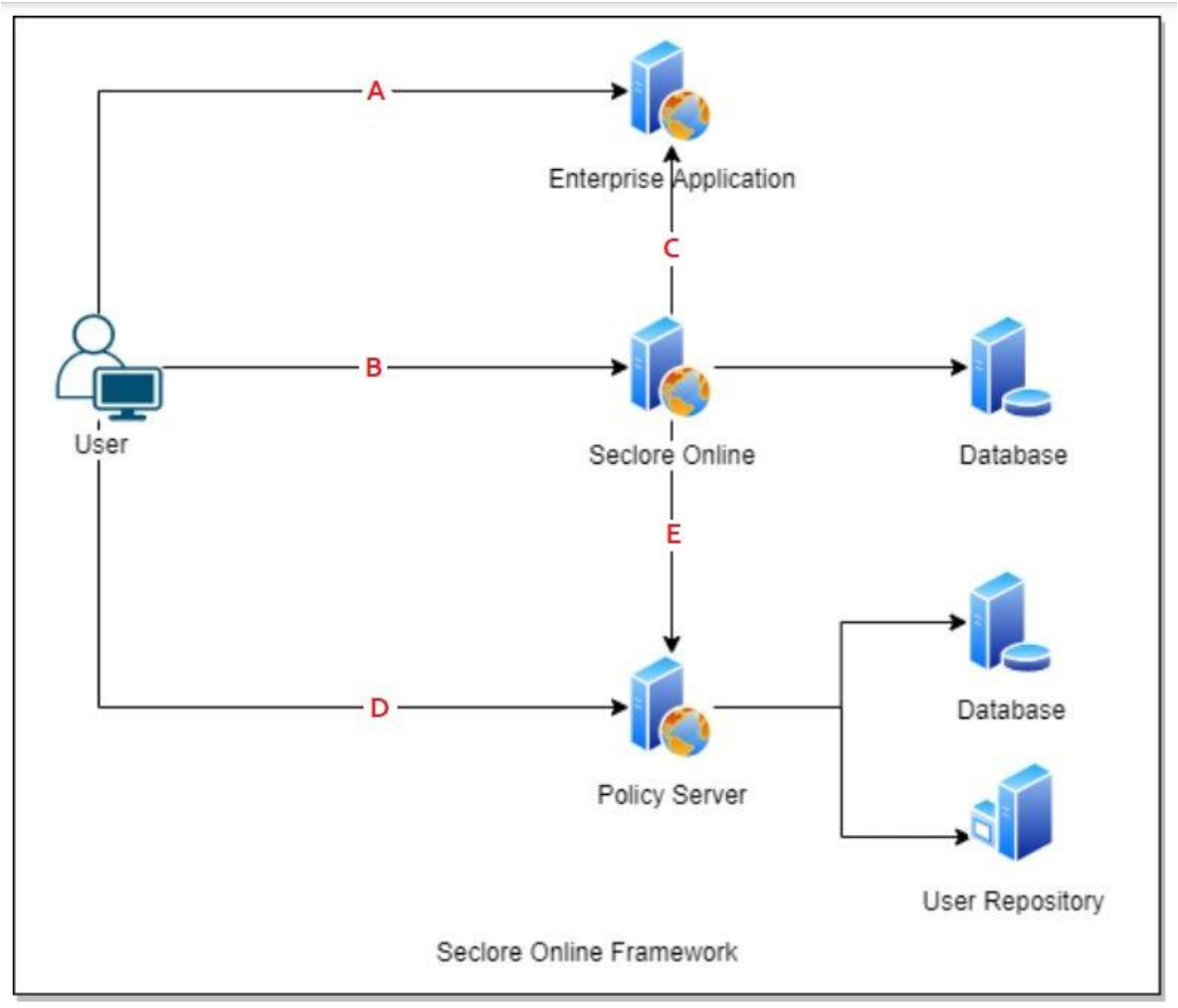
1. Functional and technical understanding of the enterprise application that is to be integrated.
2. Understanding of how data/files/content moves in and out of the enterprise application.
3. Hands on knowledge of restful web services, JSON.

1.2. About Seclore Online

Seclore Online lets users access Seclore-protected files and emails without installing any agent/software. Users can view, edit, and print the files in the browser itself. When users open a Seclore-protected file, usage permissions are enforced, and activities are tracked. This is a very simplified and rich user experience, where the user is authenticated before accessing the file. Once authenticated, the file opens in view/edit mode based on the permissions defined for the user. Once edited, the user can:

- Download an edited copy of the protected file.
- Receive a copy of the edited file over email within minutes of closing the file.
- Save the edited file back to the enterprise application.

1.3. About Seclore Online Framework



Here's how the Seclore Online Framework works

- User uses a file open option in the enterprise application to access the business data/content/file.
- Enterprise application redirects the user to Seclore Online to open the file online.
- Seclore Online fetches the file content from the enterprise application.
- Seclore Online redirects the user to Policy Server for authentication.
- Seclore Online fetches the permissions of the user on the given file from Policy Server.
- Based on the user's permissions on the file or the integration setting, Seclore Online opens the file in a browser in view/edit mode.

1.4. Seclore Online Use Cases

Let's go through some use cases where the Seclore Online integration provides data-centric security with ease-of-use.

Document Management System (DMS) / Content Collaboration Platform (CCP)

Seclore can enable automatic protection of files before they are downloaded from a DMS system. This ensures that information stays secure even outside the DMS system. Permissions applied on the downloaded files are fetched from the DMS itself (called 'Policy Federation') in real time. Thus, the security of the DMS system is extended to the downloaded file wherever it travels.

Enterprise File Sync and Share (EFSS) system

Files often need to be downloaded from EFSS systems to view and edit. But when they are downloaded, they lose their security completely. They can easily be misused. Seclore integration automatically protects the files as soon as they are downloaded from an EFSS system, thereby protecting the files wherever they go.

Virtual Data Room (VDR) systems

Files often need to be downloaded from VDR systems by users to view locally on their computers or mobile devices. But when they are downloaded, they lose their security completely. They can easily be misused or shared with unauthorized parties. Seclore protects the files as soon as they are downloaded from a VDR system, thereby protecting the files wherever they go.

In context of the use cases mentioned above, users need to download and install the Seclore agent to open the protected files. By integrating with Seclore Online, users don't need to install any agent. Thus, the data is not only protected, it's easy to access as well.

2. Seclore Online Integration Implementation Details

2.1. Key concepts of Communication structure between Enterprise Application and Seclore Online

2.1.1. File Token

A File Token is a string that represents a file being operated on. A host must issue a unique ID for any file used by the enterprise application. Seclore Online will, in turn, include the file token when making requests to the enterprise application. Thus, a host must be able to use the file token to locate a particular file.

A file Token must:

- represent a single file
- be a URL-safe string because tokens are passed in URLs

2.1.2. Access Token

An access token is a string used by the host to determine the identity and permissions of the issuer of a request. The enterprise application has the information about user permissions on the document, not Seclore Online. For this reason, the enterprise application must provide an access token that Seclore Online will pass back to it on subsequent requests. When the enterprise application receives the token, it either validates it, or responds with an appropriate HTTP status code if the token is invalid or unauthorized.

Seclore Online requires no understanding of the format or content of the token; Seclore Online includes it in requests and expects the host to validate it. However, access tokens must adhere to the following guidelines:

Access tokens must be scoped to a single user and resource combination. Seclore Online will never assume that an access token issued for a particular user/resource combination is valid for a different user/resource combination.

Access tokens must be valid for the user permissions that are provided by the host in the CheckFileInfo response. For example, if the view action is invoked, Seclore Online may re-use that token when transitioning to edit mode. Thus, Seclore Online will expect that any access token is valid for operations that the user has permissions to perform.

Access tokens should expire (become invalid) automatically after a period of time. Hosts can use the access_token_ttl property to specify when an access token expires. Seclore Online expects that an access token will remain valid until it expires (as indicated by the access_token_ttl value).

Hosts should not revoke access tokens as a standard part of their operations; tokens should only be revoked if a user's permissions have changed or been revoked.

2.1.3. Access Token Time To Live

The access_token_ttl property tells Seclore Online when an access token expires, represented as the number of milliseconds since January 1, 1970 UTC. Despite its misleading name, it does not represent a duration of time for which the access token is valid. The access_token_ttl is never used by itself; it is always attached to a specific access token.

2.1.4. File Hash

In the process of opening the file, Seclore Online fetches the file contents from the enterprise application and stores it temporarily. It also stores the file hash of the file. If an open request is sent for the file with the same file hash that Seclore Online has, then it simply uses this stored file content instead of fetching it again from the Enterprise Application.

2.1.5. Session Context

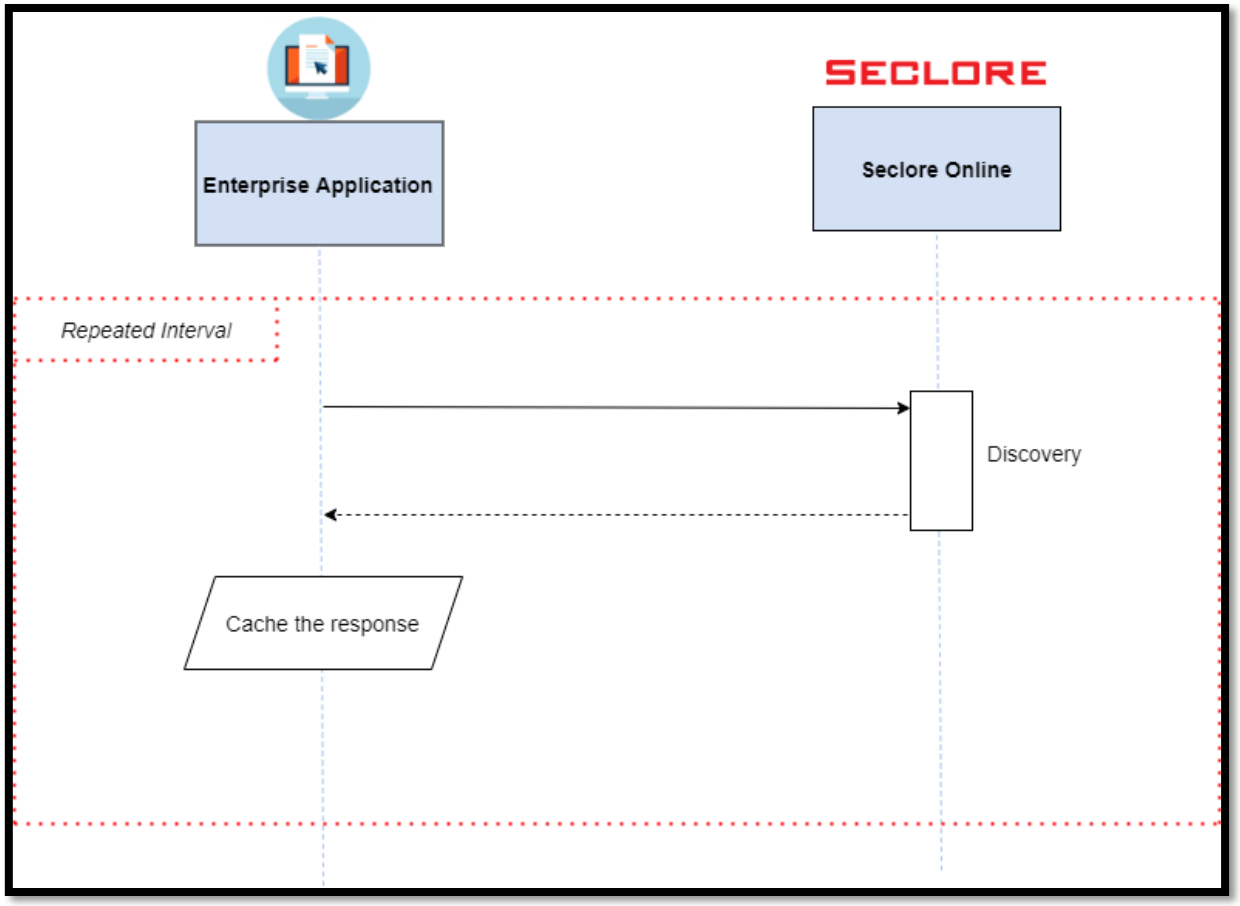
The Session Context is an optional property that the enterprise application can generate to store session related data. This data, once sent to Seclore Online, will be sent back in the subsequent requests.

2.1.6. Cloud File Access on Desktop (CFAD)

Seclore Online can also open the file natively by using the Seclore's Desktop client on the user's machine. The file once edited by the user is then saved in the Enterprise Application. More details on this can found here and technical details can be found in APIs and Technical Details.

2.1.7. Discovery

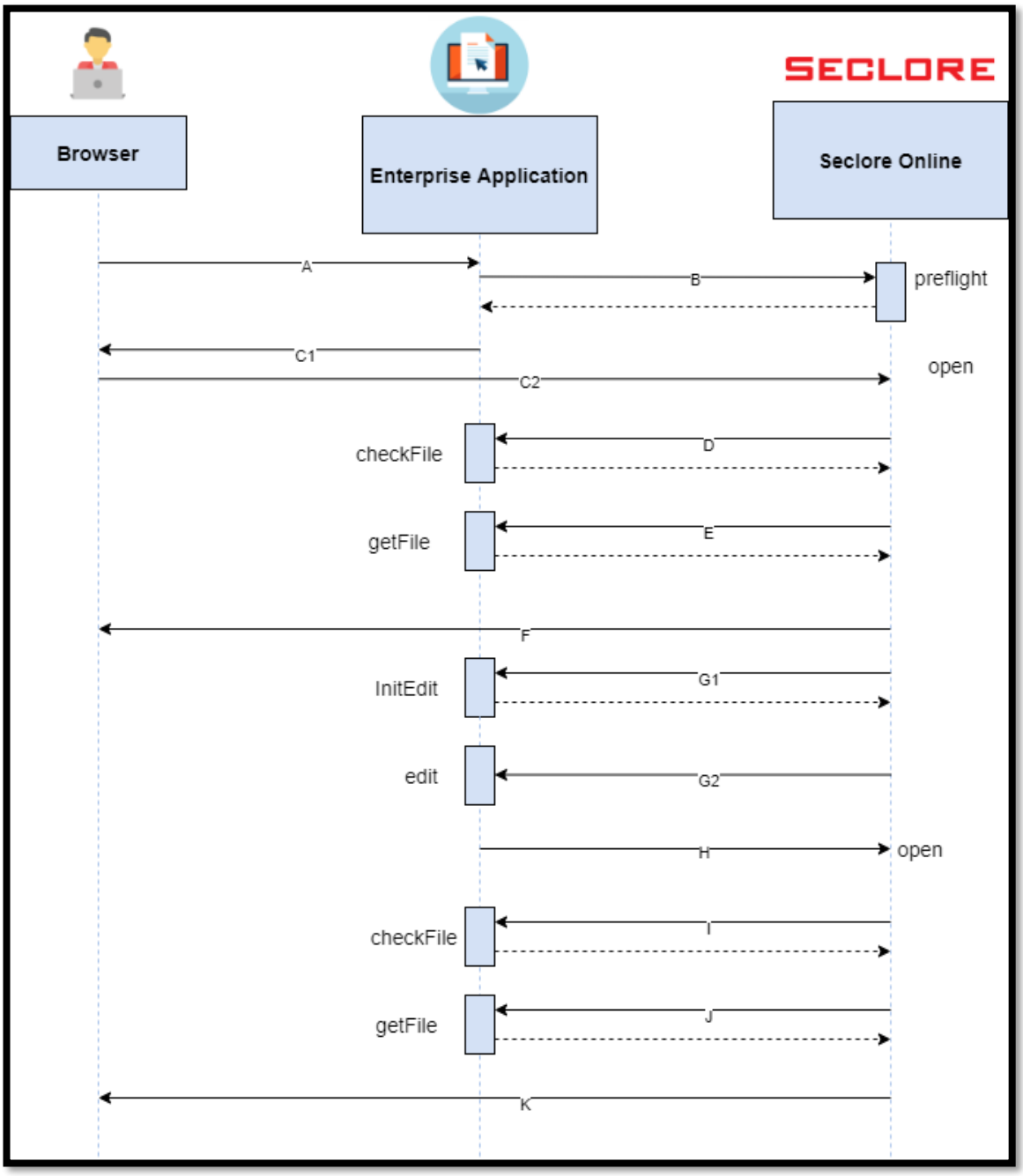
Discovery is the process by which an enterprise application determines how to interact with Seclore Online. The enterprise application should cache the discovery response. Although this discovery response does not change often, we recommend that you issue a request for the discovery periodically to ensure that you always have the most up-to-date version. 12 to 24 hours is a good cadence to refresh although in practice it is updated much less frequently.



Another more dynamic option is to re-run discovery when proof key validation fails, or when it succeeds using the old key. That implies that the keys have been rotated, so discovery should be re-run to obtain the new public key. More information can be found in the section on Integrity of Requests from Seclore Online to Enterprise Application.

2.2. Communication flow

2.2.1. File Open in view mode and switched to Edit mode

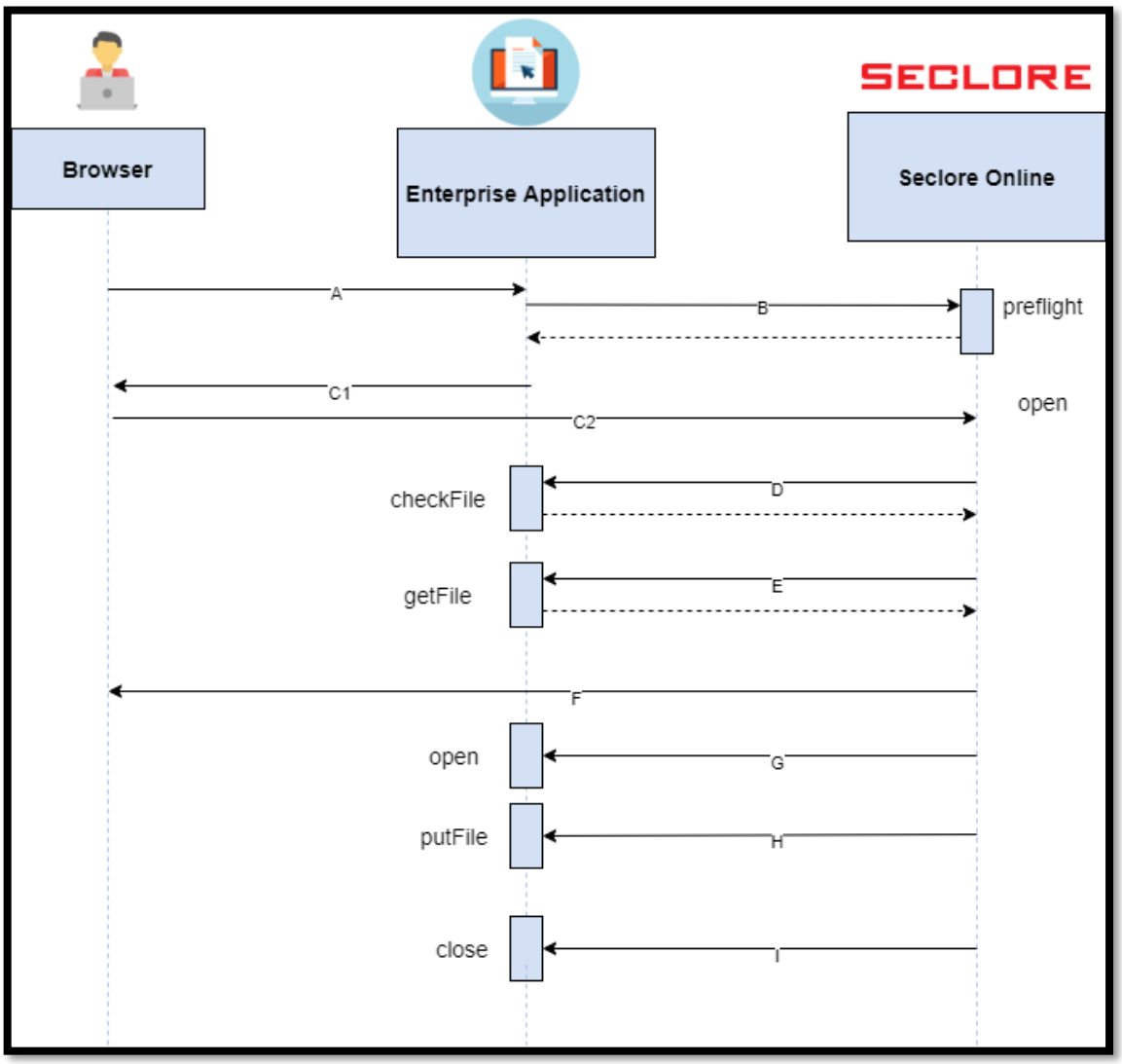


Here's how it works:

- User accesses the file via the enterprise application portal. The request goes to the enterprise application.
- The enterprise application calls the pre-flight endpoint of Seclore Online. Seclore Online checks whether the file can be opened with the file size and extension.
- After a success response from pre-flight request, the enterprise application redirects the browser to Seclore Online's open endpoint with the request details.
- Seclore Online calls the checkFile endpoint of Enterprise Application to get the file details. The enterprise application returns the 'allowed-action' field as 'view' and 'allow-edit' as 'true' to open the file in view only mode.
- After getting the details, Seclore Online calls the getFile endpoint to get the actual file contents.
- Seclore Online redirects the browser to the renderer to open the file in view mode.
- If the file is editable, the user will be able to switch to edit mode by clicking the 'edit' button in the title bar. Seclore Online will send a request to Initedit endpoint of the enterprise application to check out the file for edit. Seclore Online will then send the request to edit endpoint of enterprise application.

- The edit endpoint will call the Seclore Online's open endpoint once again to open the file.
- Seclore Online calls the checkFile endpoint of enterprise application to get the file details. The enterprise application returns the 'allowed-action' field as 'edit' to open the file in edit mode. It also returns 'allow-saveback' as true to allow for putFile.
- After getting the details, Seclore Online calls the getFile endpoint to get the actual file contents.
- Seclore Online redirects the browser to the renderer to open the file in edit mode.

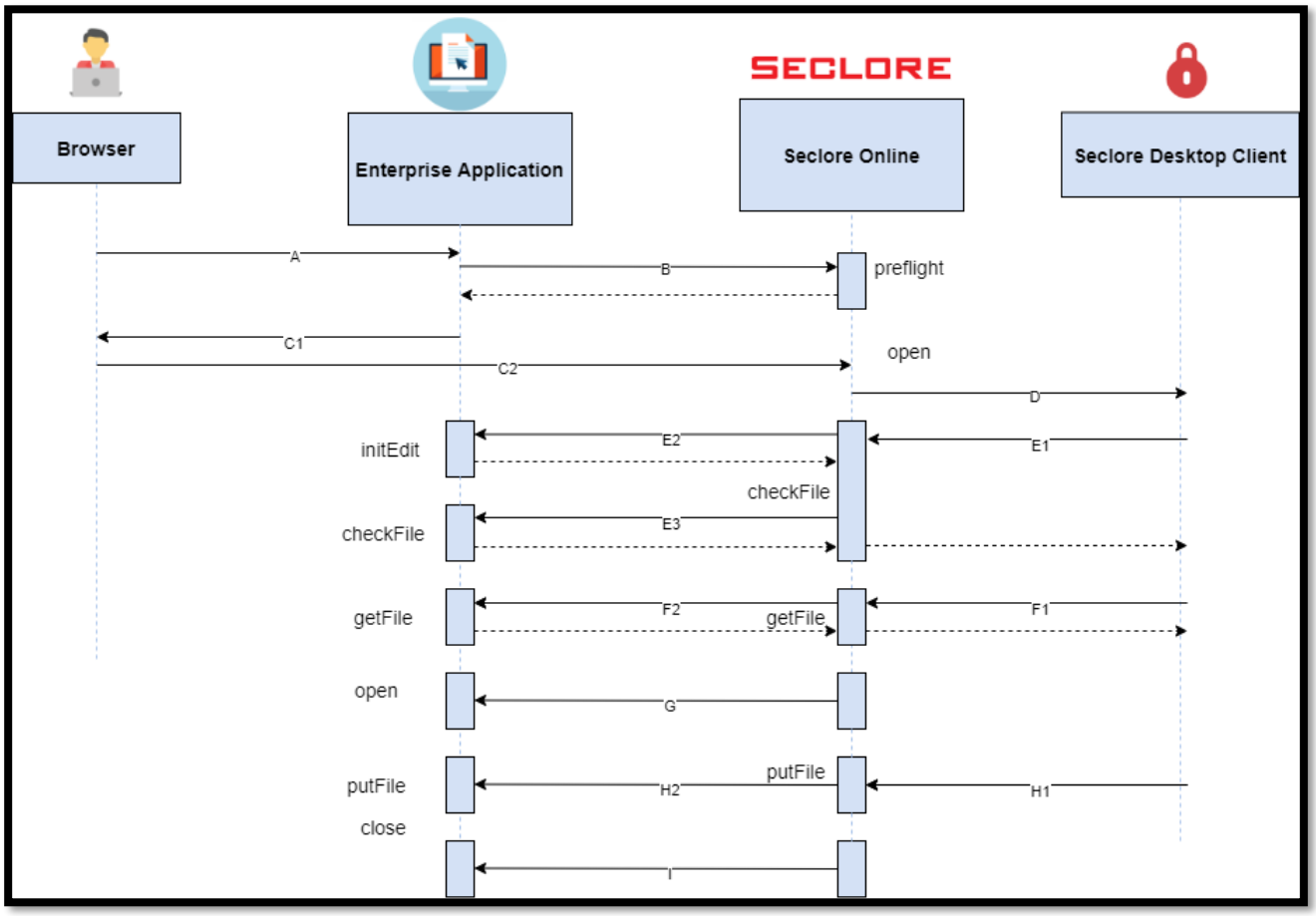
2.2.2. File Open flow with Save-back



Here's how it works:

- User accesses the file via enterprise application portal. The request goes to the enterprise application.
- Enterprise application calls the pre-flight endpoint of Seclore Online. Seclore Online checks whether the file can be opened with the file size and extension.
- After a success response from pre-flight request, the enterprise application redirects the browser to Seclore Online's open endpoint with the request details.
- Seclore Online calls the checkFile endpoint of enterprise application to get the file details.
- After getting the details, Seclore Online calls the getFile endpoint to get the actual file contents.
- Seclore Online redirects the browser to the renderer to open the file.
- An Open event is sent from Seclore Online to the enterprise application when the user opens the file in the browser.
- When the user saves the file in the browser, a putFile request is sent to the enterprise application with the file content.
- When the user closes the file, a close event is sent from Seclore Online to the enterprise application.

2.2.3. File open via Cloud File access on Desktop (CFAD)



Here's how it works:

- User accesses the file via the enterprise application portal. The request goes to the enterprise application.
- The enterprise application calls the pre-flight endpoint of Seclore Online. Seclore Online checks whether the file can be opened natively with the file size and extension.
- After a success response from pre-flight request, the enterprise application redirects the browser to Seclore Online's open endpoint with the request details.
- User is then redirected from the browser to Seclore Desktop Client.
- Desktop Client calls the checkFile endpoint of Seclore Online which in turn calls the initEdit endpoint of the enterprise application. After a successful initEdit response, Seclore Online calls the checkFile endpoint of the enterprise application to get the file details.
- After getting the details, Desktop Client calls the getFile endpoint of Seclore online which in turn calls the getFile endpoint of Enterprise application to get the actual file contents.
- Desktop Client opens the file natively on the user's machine. An open file event is sent to enterprise application from Seclore Online.
- After editing and saving the changes, the file is then sent to putFile endpoint of Seclore Online which then relays the file to putFile endpoint of the enterprise application to save it back.
- After the user closes the file on their machine, a close file event is sent to the enterprise application from Seclore Online.

2.3. Integrity of Requests from Seclore Online to Enterprise Application

2.3.1. Discovering Proof keys

Seclore Online Discovery is the process by which an enterprise application identifies proof keys, Seclore Online versions, and supported features.

The enterprise application host should cache the data. Although this data does not change often, we recommend that you issue a request for the discovery periodically to ensure that you always have the most up-to-date version.

Another option is to run discovery whenever one of the application restarts. All these approaches, as well as combinations of them, can be used.

2.3.2. Proof Validation and Rollover

When processing requests from Seclore Online for the web, you might want to verify that these requests originated from Seclore Online. To do this, you use proof keys.

Seclore Online signs every request with a private key. The corresponding public key is available in the proof-key element in the discovery. The signature is sent with every request in the X-Seclore-Proof and X-Seclore-ProofOld HTTP headers.

The signature is assembled from information that is available to the host when it processes the incoming request.

To verify that a request came from Seclore, you must:

- Create the expected value of the proof headers using the Access Token, complete URL of the endpoint and timestamp in milliseconds.
- Use the public key provided in discovery to decrypt the proof provided in the X-Seclore-Proof header.
- Verify the expected proof with the decrypted proof. If verified, the request originated from Seclore Online.

After you import the key, you can use a verification method provided by your cryptography library to verify incoming requests were signed by Seclore Online. Because Seclore Online, in future, can rotate the current and old proof keys, you have to check three combinations of proof key values:

- The X-Seclore-Proof value using the current public key
- The X-Seclore-ProofOld value using the current public key
- The X-Seclore-Proof value using the old public key

If any one of the values is valid, the request was signed by Seclore Online.

When validating proof keys, if a request is not signed properly, the host must return a 500 Internal Server Error.

The expected proof can be reconstructed using the target URL, access token, and current timestamp in the following way in Java language:

```
private static byte[] getExpectedProofBytes(String pUrl,String pAccessToken,String pTimestamp)
{
    final byte[] accessTokenBytes = pAccessToken.getBytes(StandardCharsets.UTF_8);
    final byte[] hostUrlBytes = pUrl.toUpperCase().getBytes(StandardCharsets.UTF_8);
    final Long timestamp = Long.valueOf(pTimestamp);
    final ByteBuffer byteBuffer=ByteBuffer.allocate(4 + accessTokenBytes.length +4+ hostUrlBytes.length +4+8);
    byteBuffer.putInt(accessTokenBytes.length);
    byteBuffer.put(accessTokenBytes);
    byteBuffer.putInt(hostUrlBytes.length);
    byteBuffer.put(hostUrlBytes); byteBuffer.putInt(8);
    byteBuffer.putLong(timestamp);
    return byteBuffer.array();
}
```

This expected proof can be verified with the received proof by using the following code in Java language.

```
public static boolean verifyProofKey( String strModulus, String strExponent, String strProofKey,
byte[] expectedProofArray ) throws Exception
{
    PublicKey publicKey = getPublicKey( strModulus, strExponent );
    Signature verifier = Signature.getInstance( "SHA256withRSA" );
    verifier.initVerify( publicKey );
    verifier.update( expectedProofArray ); // Or whatever interface specifies.
    final byte[] signedProof = DatatypeConverter.parseBase64Binary( strProofKey );
    return verifier.verify( signedProof );
}
```

2.3.3. Renew Access Token

The Enterprise application creates the Access Token that is sent to Seclore Online. It is also the enterprise application's responsibility to expire the token after a defined interval. Any request made with an expired Access token must return an error of 401.

Whenever a 401 – Unauthorized exception is encountered, Seclore Online will try to renew the access token by calling the Renew Access Token endpoint of the enterprise application. The enterprise application should validate the expired access token and provide a new access token. Seclore Online will then use this new value of access token for making the failed requests and use it for further requests.

3. Configurations

3.1. Configure Enterprise Application URL as whitelisted URL in Seclore Online.

To open protected files using Seclore Online, the enterprise application needs to be whitelisted with Seclore Online. Failing to do so, Seclore Online will not accept requests from the enterprise application. One Seclore Online can have multiple applications whitelisted in it to work with.

Configure the enterprise application URL by following the Seclore Online Installation Manual mentioned in the FAQs section. After doing this, you need to restart Seclore Online for the changes to take effect immediately.

4. Design Considerations

4.1. Failed Access token renewal

Access Token is used to determine the identity and permissions of the issuer of a request. The Access Token is issued by the enterprise application to Seclore Online. The Access Tokens need to have an expiry time for security. Seclore Online will include the Access Token when accessing the enterprise application's endpoints.

If the access token has expired, the enterprise application is expected to return 401 status code. Upon receiving this code, Seclore Online will hit the Renew Access Token endpoint with the expired access token in the header. The enterprise application should return a renewed access token in the response. Using this renewed access token, Seclore Online will retry the request that had failed because of the expired access token.

4.2. Error URL Framework

Seclore Online provides an error code and error message in case of an exception or an error. It also provides error-url in header. This URL endpoint can be used to show generic and some specific errors.

Seclore Online Provides contextual error pages. Error cases like File size is too large to be opened online or the file extension is not supported by Seclore Online, etc.

The Integrating Application can either show a custom error page of their own or use these Seclore Online error pages to show meaningful error pages to the user.

Information on using the Seclore Online error framework can found in section 5.2.

4.3. Version Conflict Resolution

It is possible that more than two users are trying to edit and save the file. In this scenario, the enterprise application needs to take care of version conflict. There can be many ways of handling version conflict apart from the ones listed below.

- Once any file is opened for edit mode for all other users the file is opened in view mode.
- Allow more than one user to edit the file simultaneously, on close of the file create a new file in the system with the edited copy of the file and let the user manually resolve the conflicts.

Enterprise application can decide the way in which the version conflict need to be resolved and accordingly design the view/edit and save/close technical operations.

5. APIs and Technical details

This section contains the technical details about the APIs required to communicate with Seclore Online. Since the headers are common and repeated for most of the APIs, the complete appendix is given at the end of this section.

5.1. Seclore Online Endpoints

5.1.1. Discovery

Method	GET
Path	/seclore/discovery
Description	This endpoint will return proof keys which will enable Seclore Online and the enterprise application to securely communicate.

Request Headers	
X-Request-Id	*Optional*

Success Response:

Status Codes	200

Response Headers	
Content-Type	*(Mandatory)* application/json; charset=utf-8
X-Request-Id	*Mandatory*

Response Body

```
{
        "proof-keys": {
                "old": {
                        "proof-key": {
                        "modulus": "<BASE64_ENCODE_BYTES>",
                        "exponent": " <BASE64_ENCODE_BYTES>",
                        "algo":"String"
                        }
                },
                "new": {
                        "proof-key": {
                        "modulus": "<BASE64_ENCODE_BYTES> ",
                        "exponent": "<BASE64_ENCODE_BYTES> ",
                        "algo":"String"
                        }
                }
        }
        "versions": [1.0],
        "supported-features": [1.0]
}
```

proof-keys	Proof keys contain old and new proof keys which will be used to sign the proof for secure communication.
proof-keys.old	Once Seclore Online rolls over the keys, the new key becomes the old key.
proof-keys.new	New proof keys that get generated after a fresh rollover/deployment.
Modulus	Base64 Encoded String of Modulus of the key to sign proof.
Exponent	E Base64 Encoded String Exponent of the key to sign proof.
Algo	Algorithm used to sign the proofs. Currently it is SHA256withRSA.
Versions	Versions of the APIs. Currently only 1.0 is supported.
supported-features	Supported features 1.0 indicates that the Seclore Online can open files natively (CFAD) for the user

Error Response:

Status Codes	500

Response Headers	
X-Seclore-ErrorCode	Mandatory
X-Seclore-ErrorMsg	Mandatory
X-Seclore-ErrorURL	Optional
X-Request-Id	Mandatory

5.1.2. Pre-flight check

Method	GET
Path	/seclore/1.0/files/preflight
Description	This endpoint will help check whether the file can be opened by Seclore Online both natively and online. Seclore Online checks whether the file extension and the file size are supported for opening.

Request Headers	
X-Request-Id	Optional
X-Seclore-PolicyServerURL	Mandatory

Query Parameters	
name	Full Name of the file along with the extension. If the file is HTML wrapped, the name should have *.html* extension (eg., sample.docx.html)

size	File size in bytes.
agentless	Value should be 0 to check file open natively. Value should be 1 to check file open online.

Success Response:

Status Codes	200

Response Headers	
Content-Type	*(Mandatory)* application/json; charset=utf-8
X-Request-Id	*Mandatory*

Response Body

```
{
        "allowed-action": [String]
}
```

allowed-action	Specifies whether the file can be opened in view or edit mode. Value can be either **["view"]** or **["edit"].**

Error Response:

Status Codes	500

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*
X-Seclore-ErrorURL	*Optional*
X-Request-Id	*Mandatory*

5.1.3. Open

Method	POST
Path	/seclore/1.0/files/open
Description	After the successful pre-flight, user should get redirected to this end-point via browser. This will initiate the file opening and render the file in browser or open the file natively.

Form Parameters	
accessToken	Access Token, issued by the enterprise application which will be used to access a particular file only.
accessTokenExpiry	Time since January 1st 1970 in milliseconds when the Access Token will expire.
fileToken	A unique string that represents the file which is being worked on.
serviceUrl	This is the URL of the enterprise application which is communicating with Seclore Online to open the file. This URL will be used to access the Seclore Online specific endpoints configured in the enterprise application. For instance, if the *serviceUrl* is *https://www.acemegroup.com/services*, to access checkfile endpoint, Seclore Online will generate the checkFile endpoint URL as *https://www.acemegroup.com/services/seclore/1.0/files/{fileToken}*.
agentless	Value should be 0 to open file natively. Value should be 1 to open the file online.
policyServerURL	Policy Server URL which has protected the file.
sessionContext	(Optional) Session context can be used to store session specific data by the enterprise application.
requestId	Request ID that will be used for logging purpose.

Success Response:

The file will be opened in the browser or Seclore Desktop agent, depending upon the option selected.

Error Response:

An Error page will be shown by Seclore Online.

5.2. Seclore Online Error Framework

5.2.1. Seclore Online Error page

Method	POST
Path	The complete path will be returned in the **X-Seclore-ErrorURL** header.
Description	This endpoint can be called to show an error page. The URL for this endpoint will be returned in Error Header. Even though it is an error page, Seclore Online needs access to file specific parameters in this request. Some of the error pages may have some secondary actions related to the file. For instance, in case of the file size being too large to be opened online, the error page can provide a link to download the file locally and open it in the native Seclore Application.

Form Parameters	
accessToken	Access Token, issued by the enterprise application which will be used to access a particular file only.
fileToken	A unique string that represents the file which is being worked on.
serviceURL	This is the URL of the enterprise application which is communicating with Seclore Online to open the file.
sessionContext	(*Optional*) Session context can be used to store session specific data by the Enterprise application.
requestId	Request ID that will be used for logging purpose.

5.3. Enterprise Application Endpoints

5.3.1. Check File

Method	GET
Path	/seclore/1.0/files/{fileToken}
Description	This endpoint will return the file meta-data required by Seclore Online to open the file for the user.

Request Headers	
Content-Type	(Mandatory) application/json; charset=utf-8
Authorization	Mandatory
X-Seclore-TimeStamp	Mandatory
X-Seclore-Proof	Mandatory
X-Seclore-ProofOld	Mandatory

X-Seclore-SessionContext	Optional
X-Request-Id	Optional
X-RequestingAppName	Optional
X-RequestingAppVersion	Optional

Success Response:

Status Codes	200

Response Headers	
Content-Type	*(Mandatory)* application/json; charset=utf-8
X-Seclore-SessionContext	Optional

Response Body

```
{
        "file-name":String,
        "file-hash":String,
        "file-hash-algo":String,
        "file-size":String,
        "options":{
                "allow-edit": boolean,
                "allow-download": boolean,
                "allow-unprotected-download": number,
                "email-copy":boolean,
                "allow-saveback":  boolean
        },
        "allowed-action": [String]
}
```

Note: "allow-unprotected-download" takes 0/1 as input. All other options take boolean input.

file-name	Full name of the file with extension.
file-hash	Hash computed for the file. This will be used by Seclore Online to save a copy of the file. If the file hash matches with the file hash in Seclore Online's database, then getFile will not be called to fetch the contents of the file.
file-hash-algo	Algorithm used to compute the File Hash.
file-size	Size of the file in bytes.
options	Contains the permissions required for the file editor/viewer.
options.allow-edit	Specifies whether the user can edit the file. If set to true, the edit button will be showed in view mode. This option will be ignored if "allowed-action" is "edit" since the file is already getting opened in edit mode.

options.download	Specifies whether the user can download a copy of the file.
options.allow-unprotected-download	Specifies whether the user can download an unprotected copy of the file if user has the permissions to do so. Set 0 for deny and 1 for allow.
options.email-copy	Specifies whether an edited copy of the file will be sent to the user upon finishing editing.
options.allow-saveback	Specifies whether the file can be saved back to the enterprise application after editing.
allowed-action	Value will be 'view' if the file should be opened in view mode by the user. Value will be 'edit' if the file should be opened in edit mode by the user

Note: Apart from the configurations passed on checkFile request, Seclore Online will also check Seclore Permissions on the file and the effective permission will be used. For example, if "allowed-action" is "edit" in check file but the user does not have Seclore edit permission on the file, then the file will be opened in "view" mode.

Error Response:

Status Codes	400, 401, 500
Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*
X-Seclore-ErrorURL	*Optional*

5.3.2. Get File

Method	GET
Path	/seclore/1.0/files/{fileToken}/contents
Description	This endpoint will return the actual file contents to Seclore Online in the response body.

Request Headers	
Authorization	*Mandatory*
Content-Disposition	*Mandatory*
X-Seclore-TimeStamp	*Mandatory*
X-Seclore-Proof	*Mandatory*
X-Seclore-ProofOld	*Mandatory*
X-Seclore-FileHash	Mandatory
X-Seclore-SessionContext	*Optional*
X-Request-Id	*Optional*
X-RequestingAppName	*Optional*

X-RequestingAppVersion	*Optional*

Success Response:

Status Codes	200

Response Headers	
X-Seclore-SessionContext	Optional
Content-Type	application/octet-stream
Content-Disposition	Mandatory
X-Seclore-FileName	Mandatory

Response Body: Binary File contents.

Error Response:

Status Codes	400, 401, 500

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*
X-Seclore-ErrorURL	*Optional*

5.3.3. Download File

Method	GET
Path	/seclore/1.0/files/{fileToken}/download
Description	This endpoint will return the file contents when the user wants to download a protected copy of the file from the Seclore Online error page.

Request Headers	
Authorization	*Mandatory*
Content-Disposition	*Mandatory*
X-Seclore-TimeStamp	*Mandatory*
X-Seclore-Proof	*Mandatory*

X-Seclore-ProofOld	*Mandatory*
X-Seclore-SessionContext	*Optional*
X-Request-Id	*Optional*
X-RequestingAppName	*Optional*
X-RequestingAppVersion	*Optional*

Success Response:

Status Codes	200

Response Headers	
X-Seclore-SessionContext	*Optional*
Content-Type	*application/octet-stream*
Content-Disposition	*Mandatory*
X-Seclore-FileName	*Mandatory*

Response Body: Binary File contents.

Error Response:

Status Codes	400, 401, 500

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*
X-Seclore-ErrorURL	*Optional*

5.3.4. Put File

Method	POST
Path	/seclore/1.0/files/{fileToken}/contents
Description	This endpoint will be used by Seclore Online to save back the file after the user edits it.

Request Headers	
Authorization	*Mandatory*

X-Seclore-TimeStamp	*Mandatory*
X-Seclore-Proof	*Mandatory*
X-Seclore-ProofOld	*Mandatory*
Content-Length	*Mandatory*
SAVE_EVENT_MODE	*Mandatory*
X-Seclore-SessionContext	*Optional*
X-Request-Id	*Optional*
X-RequestingAppName	*Optional*
X-RequestingAppVersion	*Optional*

Request Body: Binary File contents.

Success Response:

Status Codes	200

Response Headers	
X-Seclore-SessionContext	*Optional*
Content-Type	*application/octet-stream*

Error Response:

Status Codes	400, 401, 500

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*
X-Seclore-ErrorURL	*Optional*

5.3.5. Init Edit

Method	POST
Path	/seclore/1.0/files/{file token}/initedit
Description	This endpoint will be used by Seclore Online when the user wants to switch from read mode to edit mode. This endpoint will check whether the file can be opened in edit mode or not.

Request Headers	
Authorization	*Mandatory*
X-Seclore-TimeStamp	*Mandatory*
X-Seclore-Proof	*Mandatory*
X-Seclore-ProofOld	*Mandatory*
X-Seclore-SessionContext	*Optional*
X-Request-Id	*Optional*
X-RequestingAppName	*Optional*
X-RequestingAppVersion	*Optional*

Success Response:

Status Codes	200

Response Headers	
X-Seclore-SessionContext	*Optional*

Error Response:

Status Codes	400, 401, 500

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*
X-Seclore-ErrorURL	*Optional*

5.3.6. Edit

Method	POST
Path	/seclore/1.0/files/{fileToken}/edit
Description	This endpoint will be called by Seclore Online after a successful initedit response. The edit endpoint will then have to call the open endpoint of Seclore Online. After which the file will be opened in edit mode for the user.

Form Parameters	
accessToken	Access Token, issued by the enterprise application which will be used to access a particular file only
fileToken	A unique string that represents the file which is being worked on.
sessionContext	(*Optional*) Session context can be used to store session specific data by the enterprise application.
requestID	(*Optional*) Request ID that can be used for logging requests.

Success Response:

The file will be opened in the browser.

Error Response:

An Error page will be shown by Seclore Online.

5.3.7. Renew Access Token

Method	POST
Path	/seclore/1.0/renewToken
Description	This endpoint will be used by Seclore Online when 401 is returned as a response for a request. This endpoint will return a renewed access token.

Request Headers	
Authorization	*Mandatory*
X-Seclore-TimeStamp	*Mandatory*
X-Seclore-Proof	*Mandatory*
X-Seclore-ProofOld	*Mandatory*
X-Request-Id	*Optional*
X-RequestingAppName	*Optional*
X-RequestingAppVersion	*Optional*

Success Response:

Status Codes	200

Response Headers	
Content-Type	*(Mandatory)* application/json; charset=utf-8

Response Body

{
"access-token": String,
"access-token-ttl": Number
}

access-token	Renewed Access Token.
access-token-ttl	New expiry time since January 1st 1970 in milliseconds.

Error Response:

Status Codes	400, 401, 500

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*

5.3.8. Open event endpoint

Method	POST
Path	/seclore/1.0/files/{fileToken}/events/open
Description	This endpoint will be used by Seclore Online to send an open file event.

Request Headers	
Content-Type	*(Mandatory)* application/json; charset=utf-8
Authorization	*Mandatory*
X-Seclore-TimeStamp	*Mandatory*
X-Seclore-Proof	*andatory*
X-Seclore-ProofOld	*Mandatory*
X-Seclore-SessionContext	*Optional*
X-Request-Id	*Optional*

X-RequestingAppName	*Optional*
X-RequestingAppVersion	*Optional*

Request Body

```
{
        " actual-allowed-action ": String
}
```

actual-allowed-action	The values will be either view or edit depending on which mode the file has been opened in. Value will either be **edit** or **view**.

Success Response:

Status Codes	200

Response Headers	
X-Seclore-SessionContext	*Optional*

Error Response:

Status Codes	400, 401, 500, 501

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*

5.3.9. Close event endpoint

Method	POST
Path	/seclore/1.0/files/{fileToken}/events/close
Description	This endpoint will be used by Seclore Online to send a close file event.

Request Headers	
Content-Type	*(Mandatory)* application/json; charset=utf-8
Authorization	*Mandatory*

X-Seclore-TimeStamp	*Mandatory*
X-Seclore-Proof	*Mandatory*
X-Seclore-ProofOld	*Mandatory*
X-Seclore-SessionContext	*Optional*
X-Request-Id	*Optional*
X-RequestingAppName	*Optional*
X-RequestingAppVersion	*Optional*

Request Body	

{
" mode ": String
}	

Mode	The values will be either manual or automatic depending on which mode the file has been closed. Value will either be **manual** or **automatic.**

Success Response:

Status Codes	200

Response Headers	
X-Seclore-SessionContext	*Optional*

Error Response:

Status Codes	400, 401, 500, 501

Response Headers	
X-Seclore-ErrorCode	*Mandatory*
X-Seclore-ErrorMsg	*Mandatory*

5.4. Status Code Appendix

Status Code	Name	Description
200	OK	Request was successful.
401	Unauthorized	Unauthorized access. This happens when Access Token as expired.
404	Not found	Requested Resources not found.
500	Internal Server Error	Internal Server error.
501	Not implemented	The requested Method was not implemented.

5.5. Header Appendix

Header Key	Value Type	Description
X-Request-Id	String	This header provides the unique identifier which can be used for logging.
Authorization	String	This header provides the access token. Value should be prefixed with Bearer <accesstoken>
X-RequestingAppName	String	Name of requesting application.
X-RequestingAppVersion	String	Version of requesting application.
X-Seclore-TimeStamp	Number	An integer that represents the intervals that have elapsed between 12:00:00 midnight, January 1, 0001, UTC and the UTC time of the request.
X-Seclore-Proof	String	A string representing data signed using the new proof key shared in discovery.
X-Seclore-ProofOld	String	A string representing data signed using the old proof key shared in discovery.
X-Seclore-PolicyServerURL	String (URL)	Comma-separated string of Policy Server URL/s which has protected the file. It can also contain a single URL.
X-Seclore-SessionContext	String	Provides the session context information. This has to be passed in the request header from the enterprise application. In case, it is received in the response then Seclore Online will pass the updated values in the subsequent requests.
user-agent	String	This optional header should contain the user agent which is opening the file.
X-Seclore-FileHash	String	File Hash of the file which is sent in the body.
X-Seclore-FileName	String	Specify the file name of file.
Content-Disposition	String	Standard Http header with the file name of attachment.

Content-Length	Number	File size in bytes.
Content-Type	String	Standard Http header.
SAVE_EVENT_MODE	String	Possible values: Manual: When the user has manually saved the file. Automatic: When the file has been automatically saved.
X-Seclore-ErrorCode	Number	This header provides an error code in case any error occurs.
X-Seclore-ErrorMsg	String	This header provides an error message in case any error occurs.
X-Seclore-ErrorURL	String	This header provides error URL in case any error occurs. This URL can be used to redirect the user to a Seclore Online error screen.

6. FAQs

6.1. How to configure a new Policy Server URL?

One Seclore Online can support multiple Policy Servers, that is, one Seclore Online can open files from multiple Policy Servers. To do this, we need to whitelist the Policy Server (URL) in Seclore Online. Here's how it can be done:

- Configure the PS URL in your enterprise application to send it along the requests as a header and parameter wherever expected.
- Configure the PS URL by following the Seclore Online Installation Manual Section FAQs.

6.2. How to open Seclore Online inside an Iframe?

Seclore Online does not open documents inside an iframe, either in view mode or edit mode. This limitation in due to security reasons. You can open the document in a new tab in the same browser instance.

7. Glossary

1. **Policy Server:** The command-and-control center of the Seclore system that is used to manage usage policies, user permissions, protected files, activity logs, and other information.
2. **Seclore Online**: Seclore component to access protected files in the browser without to the need to install any local agent/software.
3. **Enterprise Application:** Enterprise applications (SAP and Transactional Systems, Document Management Systems, Content Management Systems, Enterprise File Sync and Sharing applications) integrated with Seclore to automatically apply data-centric security to downloaded files, reports, and other extracts.
4. **User:** User who interacts with enterprise application and required to view/edit the data moving outside of the application.
5. **Protected Content:** Data/Files encrypted with Seclore.
6. **Agent:** A software required to be installed on users' machine to access Seclore-protected files.