Vitrium API Guide for External Service Authentication, Authorization & SSO

For Vitrium Security v2024.13.0+ and External Service v3.0 & v3.5

Updated: June 10, 2025
Document Revision: 1.20

Table of Contents

Document Revisions

Revision Number	Reason	Date
1.00	Created revision table	January 27, 2020
1.01	Changed references of AccessPolicy to Policy	September 15, 2020
1.02	Clarified that Authenticate calls are POST in sequence diagrams	October 21, 2020
1.03	Added information about testing v3 JSON APIs	October 22, 2020
1.04	Updated Version Unique API work flow	December 21, 2020
1.05	Added section with samples of "type" workflow	March 31, 2021
1.06	Expanded on "Permissions" endpoint and "SsoLiteToken" type descriptions	May 3, 2021
1.07	Expanded on many topics	June 23, 2021
1.08	Added oAuth2 info	August 3, 2021
1.09	Reformatted text	February 11, 2022

Revision Number	Reason	Date
1.10	Updated External Service SSO configuration and OAuth SSO configuration	July 5, 2022
1.11	Added more insight into section Secondary External Service Endpoints	October 12, 2022
1.12	Added 2 new “type” (DownloadUniqueUsernameToken and DownloadProtectedUsernameToken). Added insight into “Show External Readers”. Updated Version\Unique diagram. Added 2 new Fields “Id”, “Alias” Modified 1 Field “FileName” => “Title” Updated photo for SSO & External Service Configuration	March 20, 2023
1.13	Added Sample Request Payload (Authenticate Endpoint) Added Sample Response Payload (Permissions Endpoint)	May 29, 2023
1.14	Updated External Service Overview Added “PolicyId” Added “ConcurrentUsersLimit” Two Deprecated Request Types in v3.0: “DownloadUniqueUsernameToken” “DownloadProtectedUsernameToken” Added Appendix C (External Service v3.5)	March 7, 2024
1.15	Added “DocIncludeExternalKeys” and “FolderIncludeExternalKeys”	September 6, 2024
1.16	Added “Bot Login Protection”	December 3, 2024
1.17	Corrected typos for “WatermarkTokens” and added better samples	May 8, 2025
1.18	Added a tip for Cloudflare servers and utilizing a Service Header	May 21, 2025

Revision Number	Reason	Date
1.19	Added notes about using a “token” in Version Unique API calls that can be sent to an External Service	May 29, 2025
1.20	Added notes that explain pairs of transaction types needed to download Protected PDF from the Web Viewer.	June 10, 2025

Vitrium Security Overview

Vitrium Security is a content security and digital rights management (DRM) solution for organizations who wish to control, protect, control, and track their confidential, sensitive or revenue-generating documents and images.

Input Formats Supported

Vitrium Security accepts a variety of file inputs including PDF; text and Rich Text (RTF); Microsoft Office (Word, Excel, PowerPoint); OpenOffice word processing, spreadsheets and presentations (ODT, ODS, ODP); image (JPG, TIFF, PNG, GIF, BMP, ODG); video (MP4, MKV, FLV, 3GP, AVI, TS, MOV, WMV); audio (MP3, WAV).

Secured File Outputs

When these files are uploaded into Vitrium either through a manual process, batch upload, or automated process, they are encrypted and converted to two output formats:

Protected PDF file (documents and images)	128-bit AES encryption	Viewed with Adobe Reader or Acrobat (PC or Mac desktop versions only)
Secured Web Link (documents, images, video, audio)	128-bit AES encryption	Viewed on any web browser (on any device or platform)

Getting Started

Once you obtain a set of credentials for a Vitrium account, use these to login at:

https://login.vitrium.com/
Be sure to allow cookies for this site.

We strongly recommend and encourage you to familiarize yourself with the Vitrium user interface as it will help you when you’re working with the APIs to understand the Vitrium terminology and how the various settings are applied.

Use the in-product tutorial guide to walk you through the various sections of the software or access these quick links (also available in the Help tab):

- Getting Started Guide: http://www.vitrium.com/support/pdfs/getting_started_guide.pdf

- Administrator Manual: http://www.vitrium.com/support/pdfs/administrator_manual.pdf

Vitrium Data Management

Term Definitions

- **User** is typically *your customer* who will be accessing content protected by Vitrium Security. For historical reasons harking back to Vitrium's early PDF-only content protection days, users are often referred to as **Readers**, and the two terms are synonymous in Vitrium documentation.
- **Group** is a collection of users, usually assembled to share a common access policy or set of permissions to protected content.
- **Access Policy** contains settings which determine accessibility of protected content. For example, an Access Policy can have an expiration date after which content is no longer accessible; it can restrict content access to a set of IPs, or limit the number of devices from which content can be accessed. Access Policy is sometimes referred to as **DRM Policy**, and the two terms are synonymous in Vitrium documentation.
- **Permission** is an association entity that links a user, access policy, and content. Vitrium accommodates several levels of permissions; for example, a permission can give a single user access to specific content using a particular access policy, or a permission can be given to a group to which a user belongs for all content in a given folder. While it's important to understand the concept, in the context of External Service permissions are not particularly relevant as they are implicit.

Default Configuration

A default configuration of Vitrium, whether installed or hosted, internally stores all data relevant to content protection in a Microsoft SQL Server database. Vitrium provides a web-based Admin User Interface to facilitate management of this data, and also provides a set of APIs to allow our clients to automate this administration through custom scripts or applications if desired.

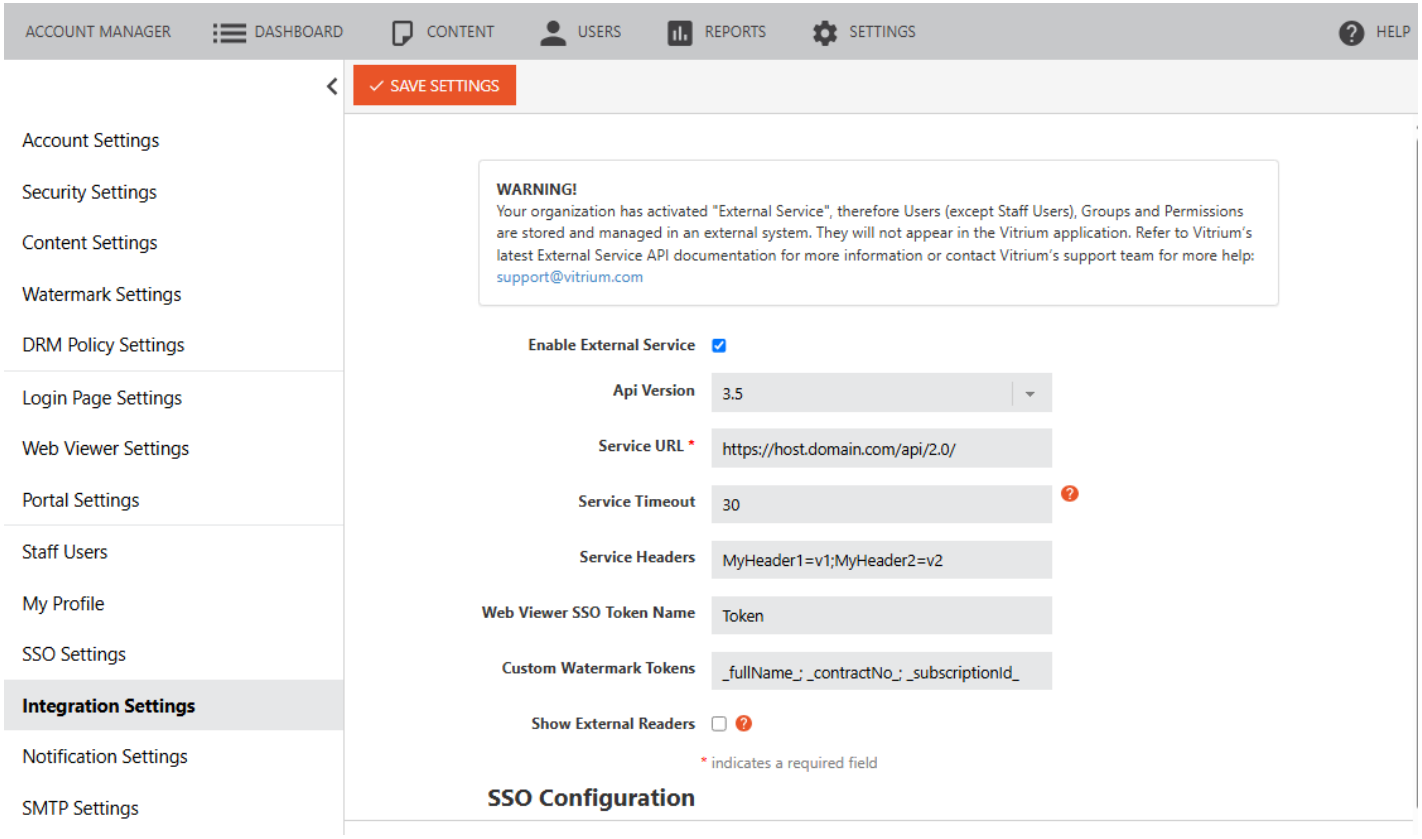
External Service Overview

To accommodate Enterprise level customers integrating Vitrium protection into existing content management systems, Vitrium implements the concept of *External Service* (also referred to as *JSON Server* or *JSON API Server*). When enabled and configured, External Service overrides Vitrium's default authorization data management: instead of using SQL Server as the data store, Vitrium retrieves user and access policy information from an external source. This external source must be a RESTful web service implementing a specific set of endpoints and accepting and returning JSON payloads, as described in this document.

It is **VERY IMPORTANT** to note that when External Service is enabled, authorization-specific entities such as users and groups **DO NOT EXIST** in Vitrium's internal data store. Instead, they are retrieved dynamically from the External Service when authorizing protected content access. However, access policies and permissions may or may not exist at the folder-level, as they are optional. When External Service is enabled, Vitrium combines these external entities with any existing folder-level policies, and the most lenient policy will be applied to the content. As such, certain Vitrium Admin UI elements are disabled or hidden when External Service is enabled. Additionally, while it is rare to require programmatically accessing Vitrium's APIs when External Service is enabled, if you choose to do so, take care to avoid managing users, groups within Vitrium's internal data store as these would create internal entities which would be ignored when authorizing content access.

Enabling External Service

Enable and configure your External Service in the Integration Settings section under Settings in Vitrium Admin UI:



Enable External Service (checkbox)

This checkbox enables/disables the use of External Service. When enabled, Vitrium routes authorization requests to your external service as described in this document. When disabled, Vitrium handles authorization requests using its SQL Server database store and internal logic.

Api Version

Select the version of external service you have implemented. **This document discusses External Service version 3.0**. Implementation details very considerably among the API versions, so make sure you consult the proper documentation if for any reason you need to implement a different version of External Service.

Service URL

The base URL of your external service, implementing the required endpoints as described in this document. Using a secured site with “https” and an SSL certificate, though not strictly required, is highly recommended. By convention, you may want to suffix your base URL with “/api/3.0/”, though this is also not a requirement. Example: “https://vitriumservice.com/api/3.0/” (trailing slash is optional).

Service Timeout

Amount of time, in seconds, that Vitrium will wait for a response from your external service. Default value is 5 seconds. It is important to make your web service as performant as possible, but under extraordinary circumstances it

may be necessary to extend this timeout. Do not increase the timeout unless there a strong reason to do so, as it may negatively affect your end users' experience.

Service Headers

Optional list of HTTP headers that Vitrium will send with each request to your web service. The headers should be specified as name=value pairs separated by semicolons (eg. "HeaderName1=HeaderValue1;HeaderName2=HeaderValue2")

The service header could be used as an additional level of security to prevent the External Service endpoints from being used by unauthorized clients.

Tip! A use case for ***Service Headers*** would be when Cloudflare servers are hosting the external service web server as Cloudflare seems to require an arbitrary user agent header. In these cases, you could add something like "User-Agent=VitriumIntegration" to this field. Multiple headers should be separated by semicolon (e.g. "User-Agent=VitriumIntegration;SecretKey=123"). Anything you configure here will be parsed and sent to your web service as part of the headers with each request.

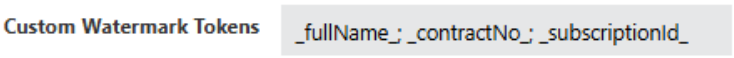
Web Viewer SSO Token Name

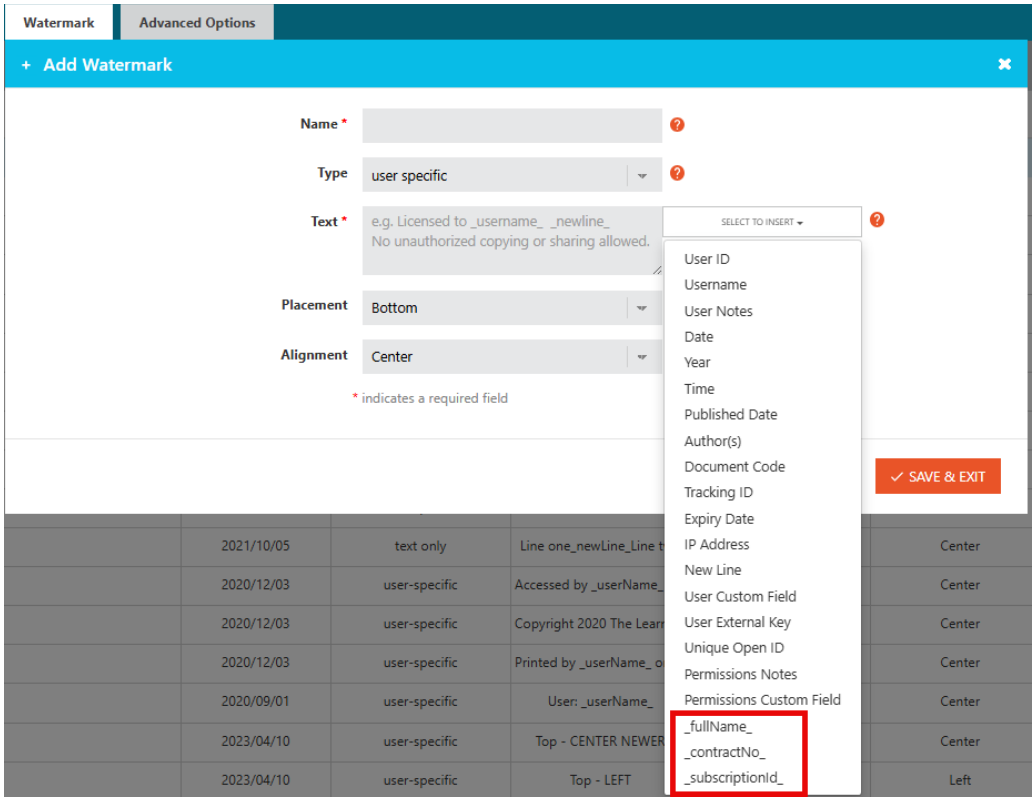
This value is only required for Single Sign-On (SSO) functionality in the Web Viewer, and sets the name of a field that is expected to be present in an unlock request and contain a token used for SSO.

For complete explanation of SSO, please refer to the section on Web Viewer SSO further along in this document.

Custom Watermark Tokens

The Custom Watermark Tokens field allows text from the Authorization response for additional watermarks. The tokens are then made available for selection in Settings -> Watermark Settings. In other words, adding the following in Settings -> Integration Settings, will present those as variables in Settings -> Watermark Settings:





Refer to **WatermarkTokens** on page 20 for details on the JSON response payload.

At document unlock time, the custom watermark tokens are replaced with values of Watermark Tokens from External Service V3.

Show External Readers (checkbox)

When enabled, you will need to ensure that your external service has a functioning Readers endpoint as documented in this guide. The endpoint will be hit when an administrator opens the USERS tab in the Admin UI.

When disabled, the recommended setting, then you do not need a Readers endpoint as the system will populate the USERS windows based on users that have past activity. This option is the most efficient.

External Service Implementation

This next section walks you through implementing your JSON-based RESTful web service.

User Authentication and Authorization

Endpoint

Unlike earlier versions of the JSON API Server, version 3 implements a single endpoint for user authentication and authorization. The endpoint must reside at the `/authenticate` route, and accepts POST requests with a JSON payload in the body containing all information necessary to retrieve a user and optionally access policy.

Request (AuthenticationRequest)

Payload

The exact contents of the JSON payload for the POST request to the `/authenticate` endpoint depend on the context at which it is invoked, but the structure is as follows:

```
{
   "Username": string (optional),
   "Id": string (optional),
   "Password": string (optional),
   "HashingKey": string (optional),
   "HashingVersion": string (optional),
   "CaseSensitivePassword": bool (optional),
   "Token": string (optional),
   "Type": string (required),
   "Document":{
      "FolderPath":[
         array of strings (GUIDs) (required),
      ],
      "DocumentId": string (GUID) (required),
      "VersionId": string (GUID) (required),
      "DocCode": string (0000-0000-000000-00000000) (required),
      "Metadata":{
         "ContentType": string (std/img/vid/aud) (required),
         "Title": string (required),
         "VersionName":string (optional),
         "UserSpecificWatermarkTemplates":[
            "Id": string (GUID) (required),
            "Name": string (required),
            "TextTemplate": string (required)
         ] (optional)
      } (required),
      "ExternalKey":string (optional),
      "Status":{
         "IsActive": bool (required),
         "IsMostRecentVersion": bool (required),
         "IsMostRecentVersionActive": bool (required),
      } (required),
      "Alias": string (required)
```

```
   } (required),
   "UserClient":{
      "AppName": string (optional),
      "AppVersion": string (optional),
      "Platform": string (optional),
      "OperatingSystem": string (optional),
      "DeviceName": string (optional),
      "DeviceId": string (optional),
      "HasOfflineAccess": bool (optional),
      "InjectVersion": string (optional),
      "IpAddress": string (optional),
      "Language": string (optional),,
      "OutOfBrowser": bool (optional),
      "ServerUrl": string (URL) (optional)
   } (optional)
}
```

Keep in mind that this payload is intended to provide an exhaustive data set intended to facilitate a great deal of flexibility in the details of your web service implementation. Do not feel daunted; you will in all likelihood only use a small subset of the information provided, customized to your specific needs.

The payload fields contain the following information:

Key	**Definition**
Type	This value will be provided with every request and will contain the type of authentication request contained in the JSON data. You may find this a useful value on which to branch your code. The field will contain one of the following values: • "UserCredentials" [2b] • "HashedUserCredentials" • "UniqueDocCopyIdToken" [1b] • "SsoLiteToken" • "WebViewerSso" • "PrintMeteringUsernameToken" • "WebViewerSessionTokenVerification" • "PhoneUnlockToken" • "DownloadUniqueUsernameToken" (only in v3.5) [1a] • "DownloadProtectedUsernameToken" (only in v3.5) [2a] [1a, 1b] A ***DownloadUniqueUsernameToken*** transaction type will be sent to the external service seeking permission for the user to download a Protected PDF from the Web Viewer and/or User Portal. When the user opens that Protected PDF, a ***UniqueDocCopyIdToken*** transaction type will be sent to the external service seeking permission for the user unlock the file and the response can with reference a DRM policy ID in the system or provide its own DRM policy within the response.

	[2a, 2b] A ***DownloadProtectedUsernameToken*** transaction type will be sent to the external service seeking permission for the user to download a Protected PDF from the Web Viewer and/or User Portal. When the user opens that Protected PDF, a ***UserCredentials*** transaction type will be sent to the external service seeking permission for the user unlock the file and the response can with reference a DRM policy ID in the system or provide its own DRM policy within the response. Refer to section ***External Service Authenticate "Type"*** on page 54 for more details on where these "type" originate. The context of this value is explained in the section ***Authorization Workflows*** that starts on page 27. Vitrium recommends reviewing ***External Service (V3) sample project code*** found on our website to obtain a better understanding of most *Type* of requests.
Username	End user attempting to gain document access. This is often, but not necessarily, the user's email address. This value is provided in all authentication contexts and scenarios and will likely be the key field used to retrieve user information from your system.
Id	
Password	Contains the password entered by a user during manual login. The password may be in plain text or hashed. You can determine the format by inspecting the value of the Type field: value of "UserCredentials" means plain text password; values of "HashedUserCredentials" means hashed password. This field will be null in token authorization contexts.
HashingKey and HashingVersion	These fields are only relevant in an obsolete context of manual login with hashed password. You are not likely to encounter these.
CaseSensitivePassword	This Boolean field is only relevant in manual login contexts. It is a property of the Content Setting associated with the protected content. The password in the password field is in lower case if CaseSensitivePassword is set to false.
Token	This is a flexible string field whose specific value depends on the authorization context, which can be determined by inspecting the value of the Type field. It may be a Web Viewer SSO token generated by your system, a Unique Document GUID, a remote unlock code generated by Vitrium for offline unlocks, or another value altogether. The specifics of this field are discussed in the following sections: ***Web Viewer SSO Token Name*** (page 11) with visual in section **WebViewerSso** (page 58)
Document	This complex type contains information about the content to which the user is attempting to gain access. This type is null if the User is logging into the portal. This type includes the following fields: *FolderPath* Lists the Vitrium IDs (GUID) of the folder(s) containing the content. This list starts at the root and proceeds through nested folders in order.

	DocumentId, VersionId, and DocCode Unique IDs used to identify content in Vitrium. DocumentId and VersionId are GUIDs uniquely identifying the content record and the version of the content, respectively. DocCode is a composite identifier in a format 0000-0000-000000-00000000 used to internally identify both the content and its version. Unless you are using a custom method for associating content in your content management system with content uploaded to and protected by Vitrium, you are not likely to use these fields. *ExternalKey* When content is uploaded and protected by Vitrium, you can associate the content with any string external key. Most commonly, this external key is the ID of the corresponding content record in your own content management system. You will most likely use this field to identify the content record, rather than the three Vitrium identifiers described above. *Metadata* Complex type describing content metadata and containing the following information: ContentType Describes the type of the content, and will be set to one of the following standardized values: • "std" for standard PDF content. These can be uploaded PDF documents, or content converted to PDF from text, Microsoft Office or OpenOffice format, such as Word documents or Excel spreadsheets. • "img" for protected images • "vid" for protected video content • "aud" for protected audio content Title and VersionName Content file name and version name in Vitrium. UserSpecificWatermarkTemplates Information related to User Specific Watermark
Status	IsActive Indicates the current status of the document. "true" if the document is active, or "false" if the document is inactive. IsMostRecentVersion Indicates whether this is the most recent version of the document. "true" if this is the most recent version of the document, or "false" if an administrator has created and uploaded a newer version of this document in Vitrium. IsMostRecentVersionActive Indicates whether the most recent version of the document is active. "true" if the document is active, or "false" if the document is inactive.
Alias	6 characters unique label for document (*case-sensitive*)

UserClient	*AppName* The name of the application used to open the document. For unlocks originating from the PDF version of the document, this contains the name of the PDF reader. For unlocks originating from the Web Viewer version of the document, this contains the name of the web browser. *AppVersion* The version number reported by the app used to open the document. *Platform* The operating system of the device used to open the document. This value is null for Web Viewer. *OperatingSystem* The operating system of the device used to open the document. This contains the name of the Desktop or Mobile operating system. *DeviceName* The type of device used to open the document. The information in it could be inferred from “Platform” and “OperatingSystem”. *DeviceId* Id to identify the device used to unlock the document. If two requests have the same DeviceId, then it is likely that the two requests come from the same device. However, since it is possible to clear a DeviceId, two requests with different DeviceId would indicate that it is likely, but not guarantee that the two requests come from different devices. *HasOfflineAccess* *InjectVersion* The version number of the PDF encryption engine used to generate the PDF document. This value is null for Web Viewer. *IpAddress* The IP address of the device used to open the document. *Language* The default language for the app used to open the document. *OutOfBrowser* “true” if the document was opened in a PDF reader. “false” if the document was opened in a browser. *ServerUrl* The internal Url that received the request to open the document.

Request Example

```
{
   "Username": "username",
   "Id": null,
   "Password": "password",
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": true,
   "Token": null,
```

```
    "Type": "UserCredentials",
    "Document":{
       "FolderPath":[
          "4a7fc427-ffb0-4080-b407-d733c65fb7ea"
       ],
      "DocumentId": "4b89a29a-a8cb-4225-a284-d3a338bc8152",
      "VersionId": "ed49afa2-6c22-4e5b-8237-04de496e0d8f",
      "DocCode": "0001-13B9-1BE6-0000209F",
       "Metadata":{
          "ContentType": "std",
          "Title": "Filename",
          "VersionName": null,
          "UserSpecificWatermarkTemplates":[{
             "Id": "9f1e1576-01b7-490b-8330-011896bf537e",
             "Name": "default",
             "TextTemplate": "Licensed to _userName_"
          }]
       },
       "ExternalKey": "ValidExternalKey",
       "Status":{
          "IsActive": true,
          "IsMostRecentVersion": true,
          "IsMostRecentVersionActive": true
       },
       "Alias": "8tGZTy"
    },
    "UserClient":{
       "AppName": "Chrome",
       "AppVersion": "83.0",
       "Platform": null,
       "OperatingSystem": "Windows 10 .",
       "DeviceName": "Other",
       "DeviceId": "WV-7065050a-6943-4233-b6d0-b344f34e02f8",
       "HasOfflineAccess": false,
       "InjectVersion": null,
       "IpAddress": "172.30.0.115",
       "Language": "en",
       "OutOfBrowser": false,
       "ServerUrl": "https://view.protectedpdf.com/api/doc/8tGZTy"
    }
}
```


Response (AuthenticationResponse)

It is your responsibility to use the data provided in the request payload by Vitrium and implement any logic necessary to decide whether the requesting user should be granted access to the protected content and what access policy should be applied. For example, you might verify that the Username supplied exists in your content management system, and use the Document.ExternalKey field to locate a content record and link it to an order for the user in your system. You might decide that the user should only have access from a single device, or for a limited amount of time; you might find that the username is invalid or that the user shouldn't be allowed access to the content at all. You must then communicate the result back to Vitrium is the JSON payload of your response.

HTTP Status

The HTTP Status of your response should *always* be **200 (OK)**.

Please implement proper error handling and refrain from returning error status codes to indicate an exceptional result, such as 404 (Not Found) for an invalid username or 400 (Bad Request) for unexpected or mismatched parameters. Vitrium treats error HTTP status codes as an indication of an incorrect implementation or server problems on your end. All authorization failures, anticipated or exceptional, should be reported in the body of the response.

Content Type

The Content type of your response should be set to **application/json**.

Payload

The JSON response payload should adhere to the following format:

```
{
  "Succeed": bool (required),
  "UserId": string (required if Succeeed is true),
  "Username": string (required if Succeeed is true),
  "Message": string (optional),
  "WatermarkTokens": dictionary (optional),
  "PolicyId": string (optional),
  "Policy": {
    "PdfLimit": int (optional),
    "BrowserLimit": int (optional),
    "ComputersMax": int (optional – mutually exclusive with PdfLimit/BrowserLimit),
    "Expiry": datetime (optional),
    "IpAddressesMax": int (optional),
    "IgnoredIpAddresses": string (optional),
    "RelativeExpiryInDays": long (optional),
    "OfflineDurationinDays": long (optional),
    "DocumentLimit": int (optional),
    "OpenLimit": int (optional),
    "PrintLimit": int (optional),
    "WebPrintLimit": int (optional),
    "LocationRestrictions": string (optional),
    "LocationPermits": string (optional),
    "AllowDownloadSourceFile": bool (optional),
    "ConcurrentUsersLimit": int (optional),
    "WebViewerDocPolicyOverride": {
      "AllowAnnotations": bool (optional),
      "AllowCopy": bool (optional),
      "AllowPrint": bool (optional),
      "AllowWebPrint": bool (optional),
      "DisableBookmarks": bool (optional),
      "DisableSearch": bool (optional),
      } (optional)
  } (optional)
}
```

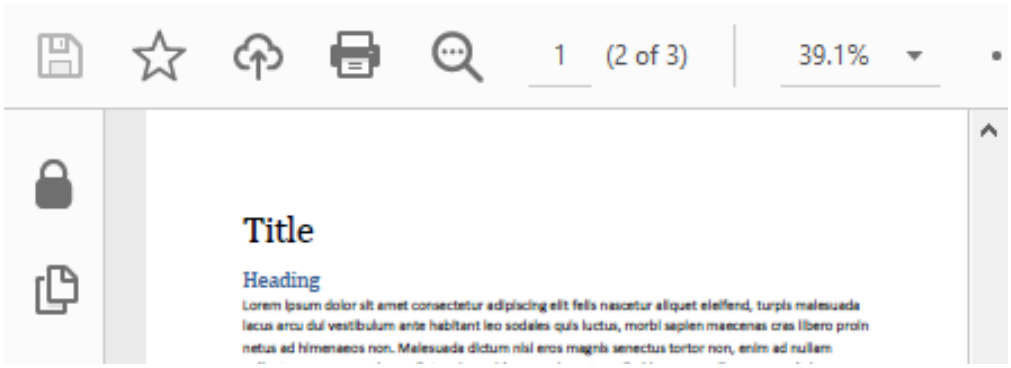
Key	Definition
Succeed	The key field in the response, this is a required Boolean value indicating whether authorization succeeded or failed.
UserId and Username	Strings uniquely identifying a user. Ideally, UserId would be a unique ID of the user record in your system – usually an integer or GUID – and Username would be a unique name or email address. Remember that with External Service enabled, Vitrium does not store user entities internally; however, Vitrium *does* use these UserId and Username identifiers to track past activity for a given user in order to ensure proper access policy validation. These two fields are **required** if Succeed is true, otherwise they are ignored.
Message	Use this field to provide any authorization failure information you want displayed to the user. Vitrium will ignore this field when Succeed is true. When Succeed is false and this field is empty or null, Vitrium will present the user with its generic "Your login credentials appear to be incorrect" message. Use this field to override this message with any custom information you wish to present to your end user instead.
WatermarkTokens	This field can be used to dynamically customize the content watermark. Note that Vitrium already provides a number of built-in dynamic watermark tokens, such as username, access expiry date, unlock date and time etc, and you should review these before adding new custom tokens. This functionality is intended primarily to include information specific to your external system and not tracked in Vitrium. In order to take advantage of this functionality, create a watermark containing a custom token, include the watermark in the Vitrium protected content, then provide the token value in this field. For example, you could create a dynamic watermark such as "Licensed to _fullName_ under Contract No _contractNo_ and Subscription ID _subscriptionId_" (note that by convention, watermark tokens in Vitrium are surrounded by underscore characters, and while this is not strictly necessary, we encourage you to follow this convention for the sake of consistency). In this example, _userName_ is one of the built-in tokens provided by Vitrium, but _contractNo_ and _subscriptionId_ are custom tokens. You could then provide values for these tokens dynamically in your response like this: `"WatermarkTokens": {"_fullName_": "John Doe","_contractNo_": "CTR123", "_subscriptionId": "SUB321"}`. Note also that there is no harm in providing extraneous tokens in your response; if no matching tokens are found in the content's watermark, the WatermarkTokens you provide will simply be ignored.
Policy	The access policy identifies the limits and conditions controlling user's access to the protected content. You specify the access policy, and Vitrium tracks all past activity and enforces it. PdfLimit, BrowserLimit and ComputersMax **PdfLimit** specifies an optional limit on the number of separate devices on which downloaded PDF content can be opened, while **BrowserLimit** is an optional limit on the number of separate browsers on which content can be opened in Web Viewer.

	If omitted or set to null, the value is Unlimited. /// Value of -1 is equivalent to Not Set and will fail validation. Expiry, RelativeExpiryInDays and OfflineDurationInDays IpAddressesMax and IgnoredIpAddresses DocumentLimit OpenLimit PrintLimit and WebPrintLimit LocationRestrictions and LocationPermits AllowDownloadSourceFile WebViewerDocPolicyOverride *AllowAnnotations* *AllowCopy* *AllowPrint and AllowWebPrint* *DisableBookmarks and DisableSearch* ConcurrentUsersLimit maximum number of users who can access protected content simultaneously
PolicyId	Instead of specifying policies using the 'Policy' field, you can utilize the 'PolicyId' field to provide the GUID string of the existing DRM policy within the Vitrium UI. This allows you to assign the corresponding policy efficiently.

Successful Response Example

```
HTTP error 200 – OK with JSON in body
JSON example 1 with successful authorization:
{
       "Succeed": true,
           "Policy": {
               "ComputersMax": 2,
               "IpAddressesMax": null,
               "IpRangeAllowed": null,
               "OfflineDurationinDays": 7,
               "Expiry": "1900-01-01",
               "DocumentLimit": null,
               "RelativeExpiryInDays": null,
               "OpenLimit": null
       }
}
```

PDF:

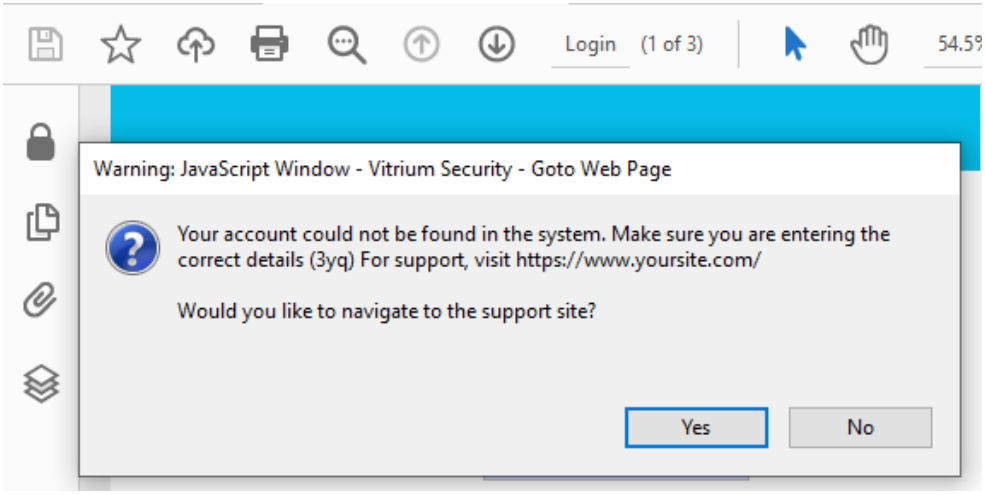


Web Viewer:

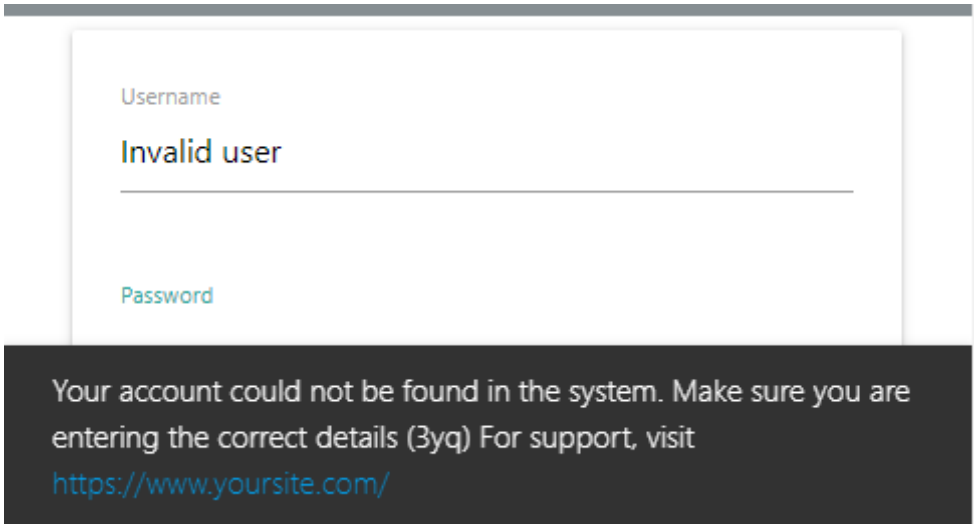
Failed Response Example

```
HTTP error 200 – OK with JSON in body JSON
example 2 with failed authorization: {
       "Succeed": false,
       "Message": "Optional. Only enabled if Succeed is false. User-specific error message string",
"Policy": null
}
```

PDF:



Web Viewer:



Permissions

Endpoint

The endpoint resides at the `/permissions` route and accepts a GET request with userid as a query parameter. **The Permissions endpoint is only used/required when the Vitrium User Portal is enabled** and its purpose is to specify what content should be presented to and therefore viewed by the user. The response needs to address the question, "What content do I want this user to see in the User Portal?" The content "filter" is accomplished by a JSON response with arrays of DocIds, FolderIds, DocExternalKeys, and/or FolderExternalKeys as noted below. In most cases, you would be returning a list of doc or folder External Keys as those would represent the product codes of the content in question.

Request

Request Example

GET http://vitriumservice.com/api/3.0/permissions?userid={userid}

Response

Payload

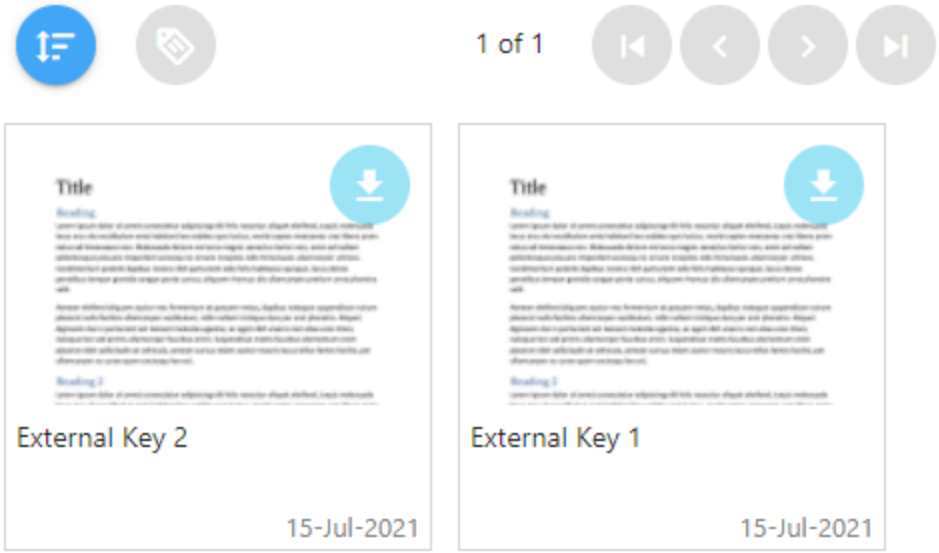
```
{
  "DocIds": Array of string,
  "FolderIds": Array of string,
  "DocExternalKeys": ["ExternalKey1", "ExternalKey2"],
  "FolderExternalKeys": Array of string
}
```

Key	Definition
DocIds	User shall have access to all Documents with the specified IDs.
FolderIds	User shall have access to all Documents contained in the specified Folders
DocExternalKeys	User shall have access to all Documents with the specified External Key.
FolderExternalKeys	User shall have access to all Documents contained in Folders with the specified External Key.

Successful Response Example


```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": ["ExternalKey1", "ExternalKey2"],
  "FolderExternalKeys": []
}
```

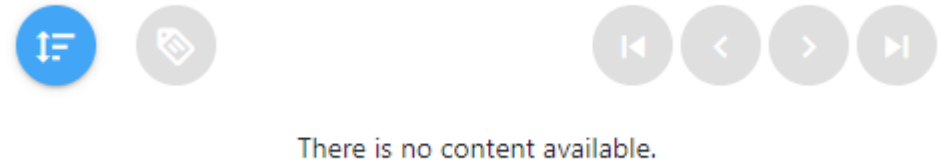

Your user sees this web portal



Failed Response Example


```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": [],
  "FolderExternalKeys": []
}
```


Your user sees a blank web portal



Readers

Endpoint

The endpoint resides at the `/readers` route and accepts a GET request with Page, Filter, and Sort as optional parameters. The endpoint is used to populate the Users tab in the Vitrium Administrative Interface.

Request

Request Examples

GET https://vitriumservice.com/api/3.0/readers
GET https://vitriumservice.com/api/3.0/readers?page={"index":1,"size":20}&sort={"Username":1}
GET https://vitriumservice.com/api/3.0/readers?page={"index":1,"size":20}
&filter={"contains":"searchterm"}&sort={"username":-1}

Key	Definition	
Page	index	Page number (first page is one, not zero).
	size	Number of items per page (default is 20)
Filter	contains	Search filter
Sort	username	Order result by the username in the following order: For Ascending order, use 1 For Descending order, use -1

Response

Payload

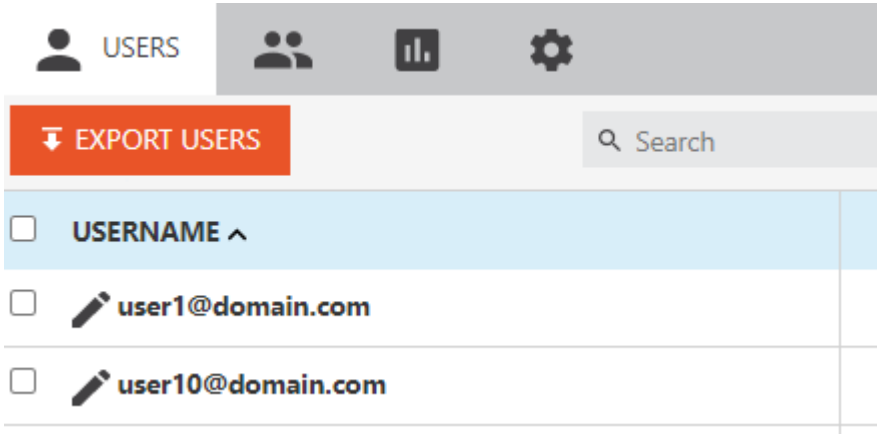
Key	Definition
Results	List of Readers
TotalRecords	Number of records that match the filter

Key	Definition
Id	ReaderId
Username	Username
IsActive	True if the reader is active, False if the user is inactive. This only affects how a reader is displayed in the user interface.

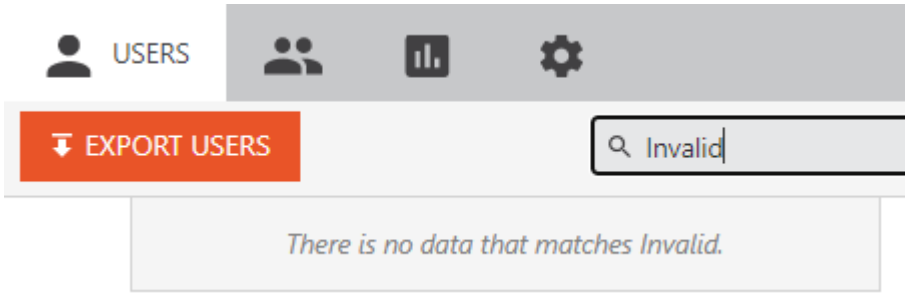
Response Example

```
HTTP error 200 – OK with JSON in body

JSON example 1 with only 1 returned item:

{
  "Results": [{
    "Id": "8d241bea-e714-4182-8257-01abfebff133",
    "Username": "name@domain.com",
    "IsActive": true,
    "AccessPolicy": null
  }],
  "TotalRecords": 1
}

JSON example 2 with more than 1 page of returned items (20 items per page):

{
  "Results": [{
    "Id": "8d241bea-e714-4182-8257-01abfebff133",
    "Username": "name@domain.com",
    "IsActive": true,
    "AccessPolicy": null
    }
  },
  …
  ],
  "TotalRecords": 25
}
```

Your Administrator sees this




```
JSON example 3 with no returned items:
{
  "Results": [],
  "TotalRecords": 0
}
```



Authorization Workflows

This section describes the different workflows and contexts under which an end user may request access to Vitrium protected content. The context is generally determined by the value of the Type field in the request sent by Vitrium to your web service's `/authenticate` endpoint. Each of the subsections below describes the circumstances of a particular context or workflow, along with how to identify and handle it in your code.

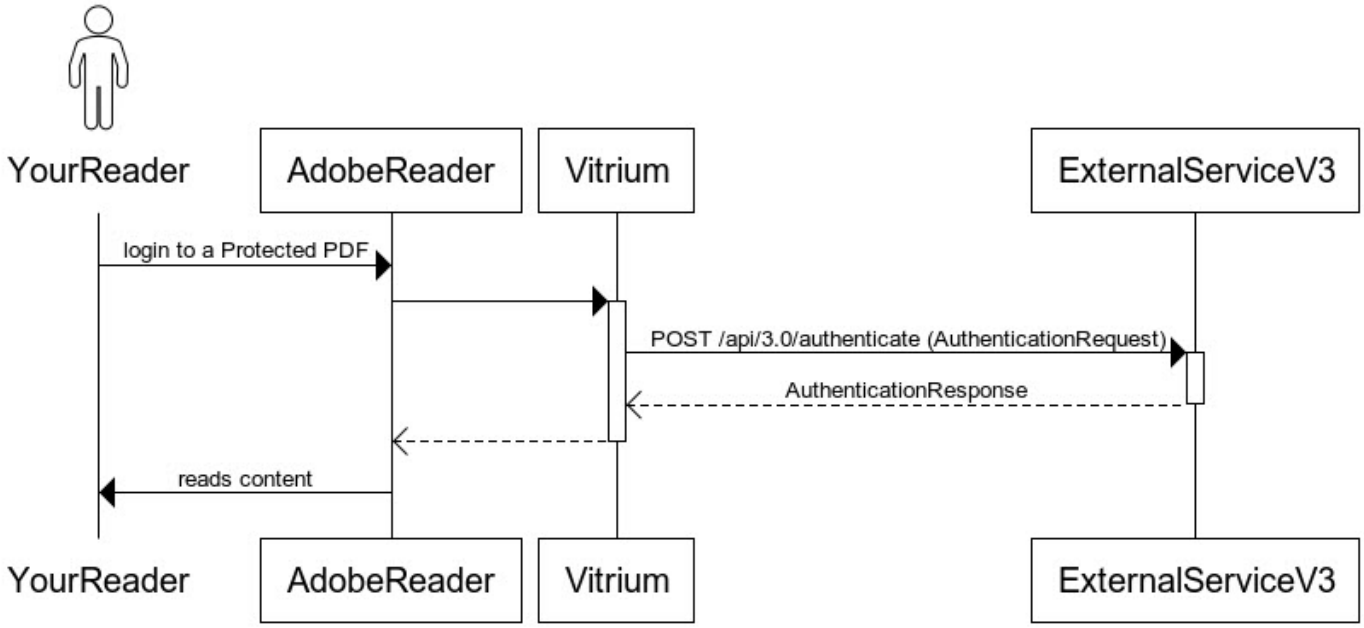
Manual Unlock

There are two different workflows for Manual Unlocks. They are:

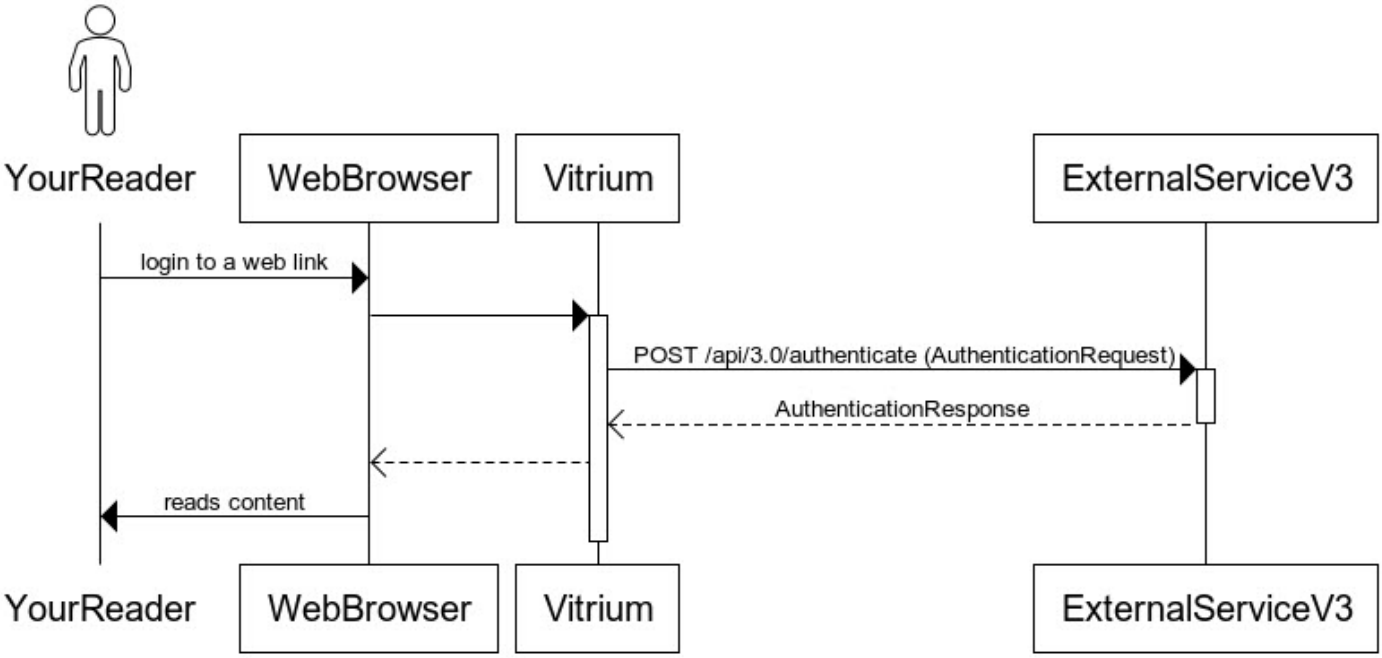
Type	Description
UserCredentials	For all new implementations for both PDF and Web Viewer
HashedUserCredentials	For backward compatibility with some older (Before 2013) PDF only; Not relevant for Web Viewer

Vitrium accounts created before 2013 may have their PDF configured to send through "**HashedUserCredentials"**. For accounts created after 2013, you will not encounter Authenticate with its Type field set to **"HashedUserCredentials"**. Please contact our support team for options if you encounter this scenario.

Reader Unlocks a PDF (With External Service V3)



Reader Unlocks Web Link (With External Service V3)



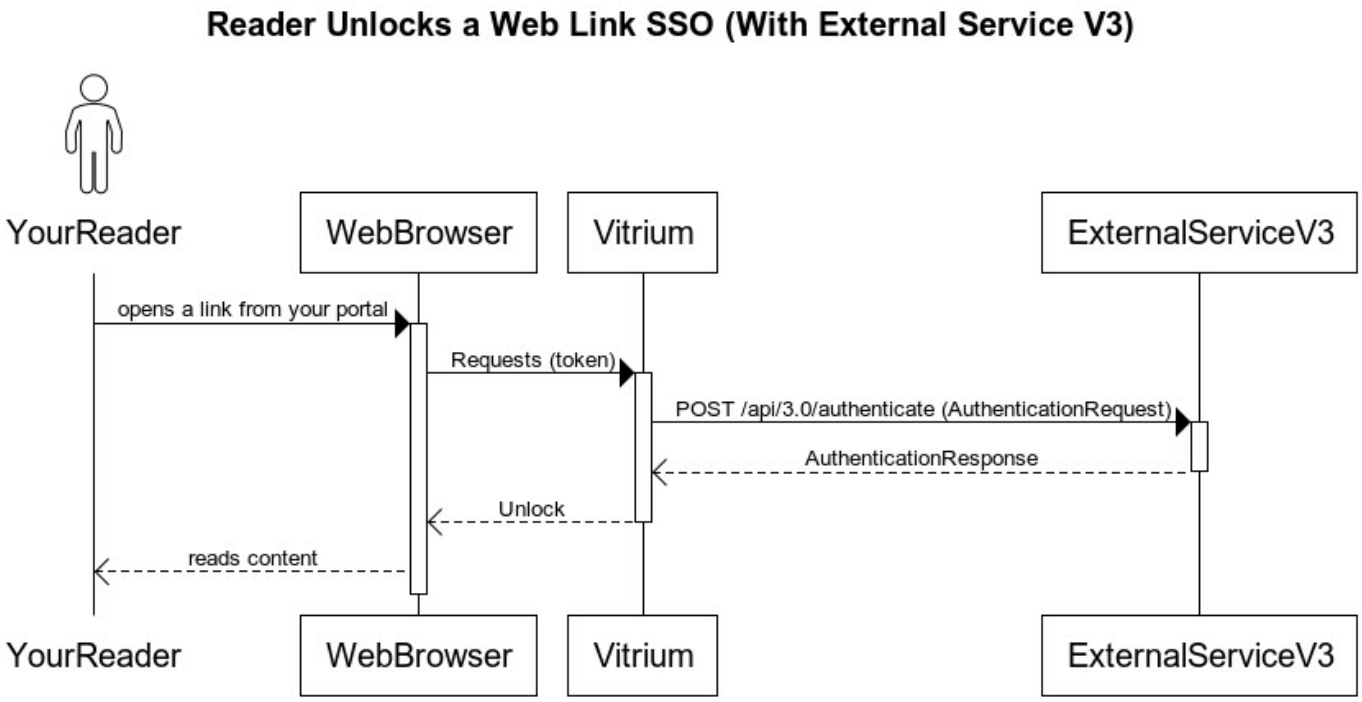
Single Sign-on (SSO) Unlock

Single Sign-on (SSO) Unlock is a more advanced workflow that builds on top of the implementation work done for the Manual Unlock workflow. You should have the Manual Unlock workflow working before starting this.

Web Viewer by Token

When properly configured, a Web Viewer SSO authorization request will have its Type field set to **"WebViewerSso"** and include an arbitrary token of your choosing which you can use to seamlessly authenticate a user and authorize their content access request.

Web Viewer by Token Workflow

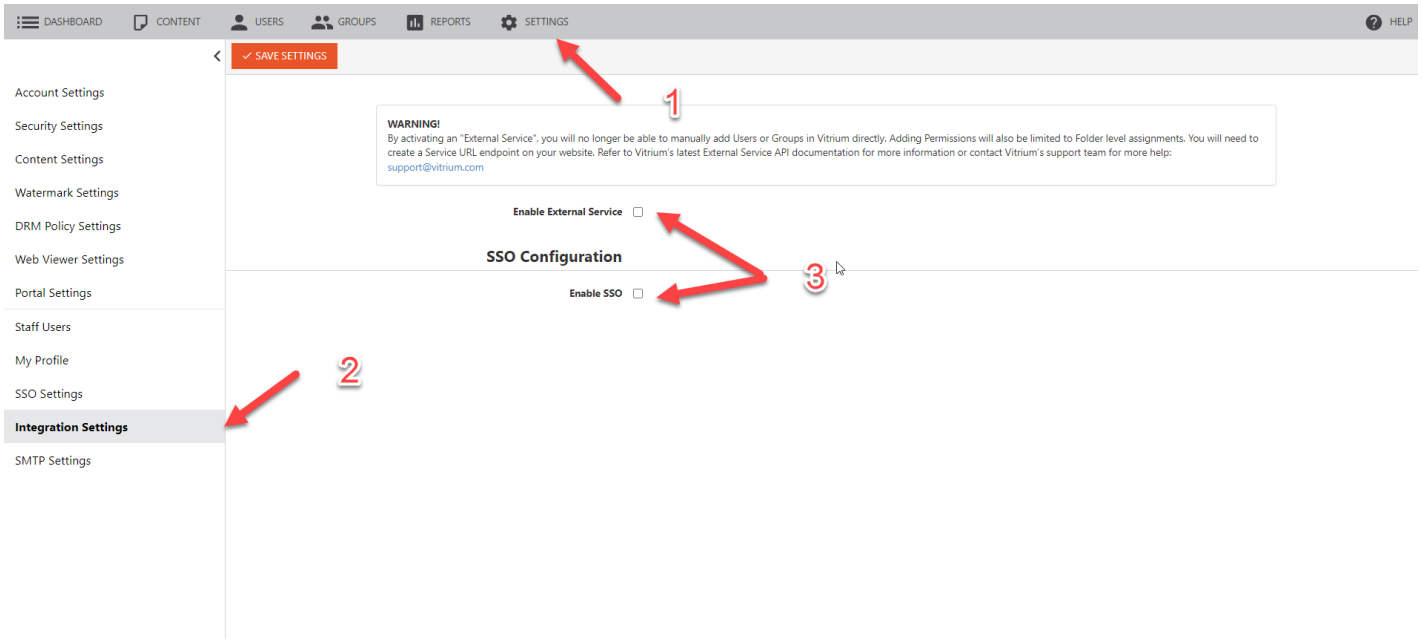


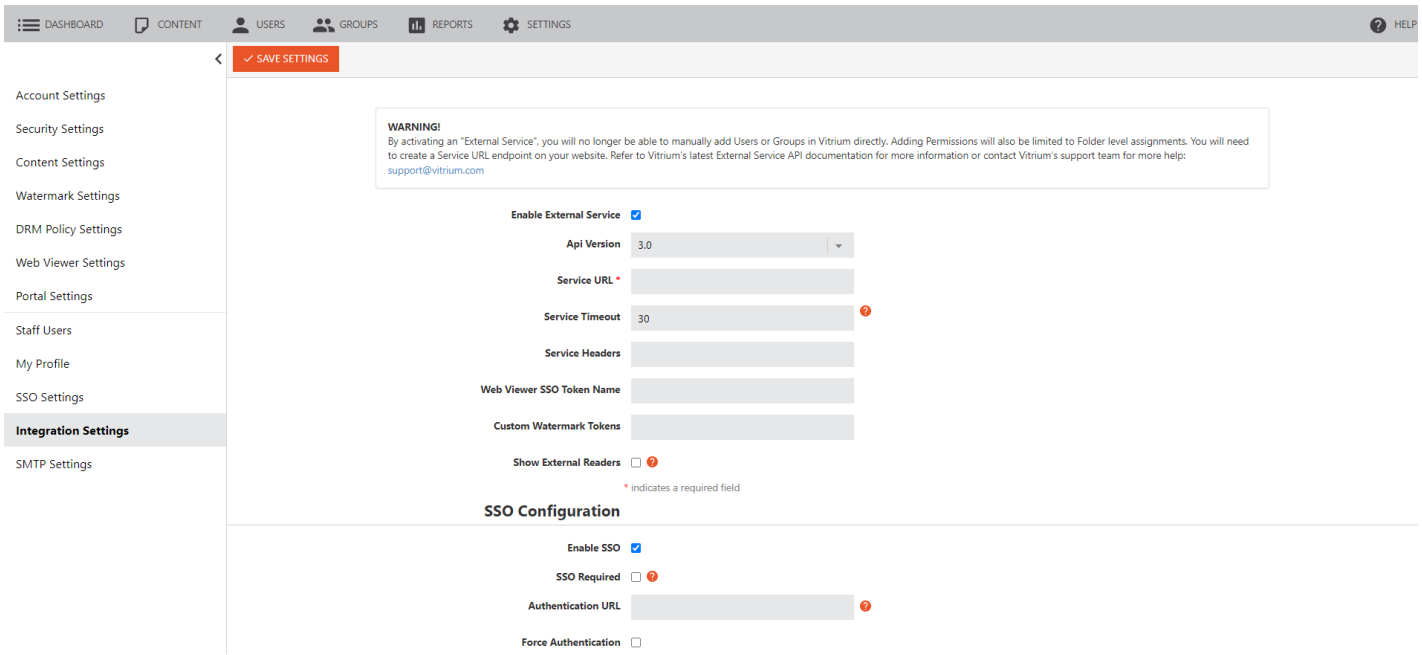
Vitrium Account Settings Configuration

The name of the parameter needs to be entered into Settings -> Account Settings -> Web Viewer SSO Token Name field in order for the Auth component to forward the fields to External Service V3

Vitrium External Settings Configuration

In order to take advantage of Web Viewer SSO functionality you first need to configure settings in the External Service Settings page of the Vitrium Admin UI. The next few pages describe most settings.





Name

Account unique SSO setting configuration name.

SSO Fail Redirection: SSO Required, Authentication URL, Authentication from Query Parameter, Skip Failed Page, and Failed Page Button Text

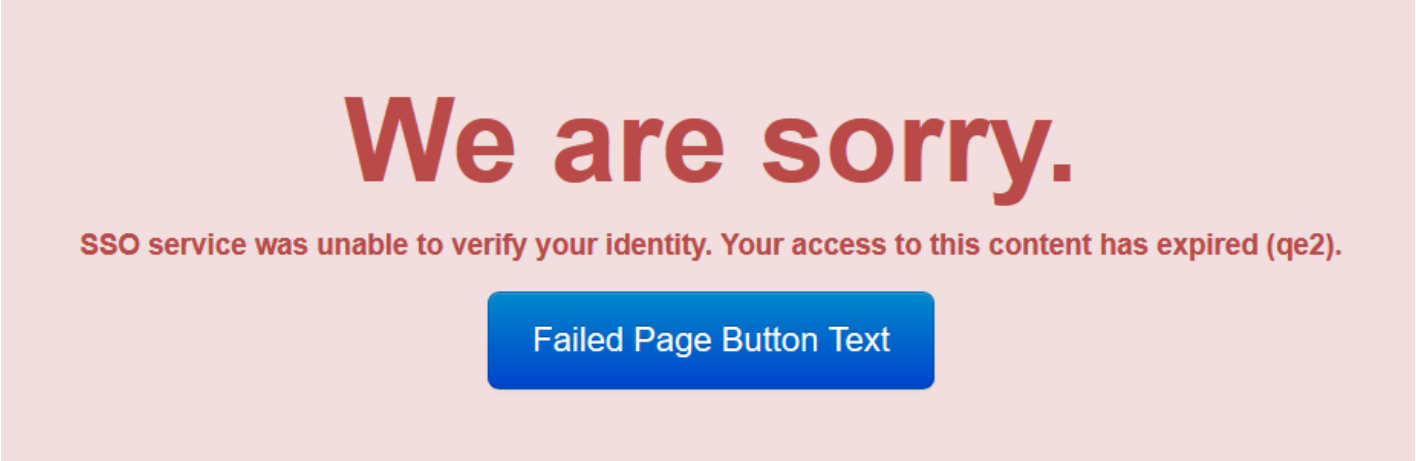
These settings combine to determine what happens if SSO fails. In this context, SSO failure could mean the following situations where authentication or authorization failed:

- Your External Service was not implemented properly
- Your External Service decided that the token provided was invalid or had expired

- Your External Service decided that the user does not have access to the document
- Your External Service provided a policy that prevents user from accessing the content. Eg. Expiry date in the past.

Basically, it covers any situation where the user was not granted access to Vitrium protected content for any reason; i.e. any result other than successful authentication and authorization.

If SSO fails, the user is redirected to the SSO Fail Page informing the user about SSO failure. For example, if your External Service returns a policy with an Expiry date in the past, the user might be presented with a page like this:



If the **Skip Failed Page** box is checked, Vitrium will skip displaying this page and instead redirect the user immediately to **Authentication URL** or the manual login page.

If the **Skip Failed Page** box is unchecked and the page is being displayed, you have the ability to customize the text of the redirect button using the **Failed Page Button Text** field as in the example above.

The **Authentication URL** contains the redirect destination in case of SSO failure. You would normally populate this field with your content management system login page URL, but if you prefer you could instead use a custom error page suggesting an appropriate course of action or any other destination. As such, this URL is a bit of a misnomer; note that in "OAuth" SSO type, this is normally the URL of the `/authorize` OAuth identity provider endpoint, and that is the origin of the name.

If **Authentication from Query Parameter** is populated, then a query string parameter with that name is appended to Authentication URL, and its value is set to the current Web Viewer content URL with any query parameters stripped off. For example, Authentication URL https://myservice.com/login, Authentication from Query Parameter "redirect_from", and failed SSO attempt to open content at https://view.protectedpdf.com /content?token=ssosecret would result in a redirection target of https://myservice.com/login?redirect_from=https%3A%2F%2Fview.protectedpdf.com%2Fcontent%3Ftoken%3Dssosecret

If **Authentication URL** is populated, it is always used as the redirection target in case of SSO failure, but the field is not required. If you leave it empty, behaviour will be determined by the other related settings. If **SSO Required** is checked,

SSO failure results in an error page without any recourse for the end user; External Service SSO is the only way that access to protected content can be granted.

If this box is unchecked and Authentication URL is empty, user will be redirected to the default Vitrium login page on SSO failure. Here they will be able to enter their username and password, and these credentials will then be passed to your External Service as part of a manual unlock authorization request.

Query Parameters

This specifies the names of the Query Parameters read by the Web Viewer and passed to the External Service. The External Service parses the parameters and based on them, identifies and validates the user.

Use commas to separate multiple Query Parameters.

Cookie Names

Similar to the Query Parameters, except that this specifies the names of the cookies read by the Web Viewer and passed to the External Service. The External Service parses the parameters and based on them, identifies and validates the user.

Use commas to separate multiple Cookie Names.

Enable Bot Login Protection

The **"Enable Bot Login Protection"** feature is designed to safeguard your system from automated scans or bots that could inadvertently consume the device limit allocated for users. This functionality ensures that only legitimate users can access your content without exceeding the device usage threshold.

Enabling this feature mitigates the risk of unauthorized device consumption, which could lead to users encountering the VC3 Device Limit Error and being unable to access the content.

1. **Verification Prompt**: When a user accesses a shared link, they will be prompted to complete a simple verification by clicking the "I'm not a robot" button.
2. **Caching Mechanism**: Once the verification is successfully completed, the confirmation will be cached for the specific device. Subsequent access from the same device will not trigger the verification prompt again.

Hide logout button in WV

Checking this checkbox hides the Logout button in Vitrium's Web Viewer.

This option is useful if you do not want users to be able to log out of Vitrium.

During Logout, the web viewer clears Vitrium's own session cookie and your SSO cookie (Referenced in the "Cookie Name" textbox), if it is used. If the SSO Cookie is also used by your web portal for other purposes, then clearing the SSO cookie could also log your user out of your portal. The Hide logout checkbox in WV should be used if this behavior is undesirable.

After logout URL

Optional URL to which the user should be redirected after clicking the Logout button in Web Viewer. This option is only available if the logout button is not being hidden by the "Hide Logout Button in WV" option.

This option should be used if your External Service does not validate username/password, or if you wish for them to use your own portal's login page.

If not set, the default behaviour is to redirect the user to the Web Viewer login form where the user could login manually.

Disable Session Interval Validation

As long as Web Viewer content is on screen, Vitrium periodically verifies that the logged in user still has access to the content. This ensures that your content is protected if you deactivate the user's access and the user is currently viewing the content. When access is deactivated, the validation process fails which locks the user out of the web viewer, erases content from offline storage, and notifies the user that he/she does not have access to the content.

This setting enables or disables this validation check.

Session Validation Interval

This setting configures the interval in which the validation check above is performed. The default is 90 minutes. When session token expires, the system will ping the external service to establish a new session.

Disable Document Ticket Interval Validation

As long as Web Viewer content is on screen, Vitrium periodically verifies that the logged in user still has access to the content. This ensures that your content is protected if you revoke the user's access and the user is currently viewing the content.

If validation is enabled, then it checks whether that the user still has access to the content. If the validation process fails, then the web viewer locks the user out of the web viewer, erases content from offline storage, and notifies the user that he/she does not have access to the content.

If validation is disabled, then the above does not occur.

Document Ticket Validation Interval

This setting configures the interval in minutes in which the above *Disable Document Ticket Interval Validation* check is performed. The default is 5 minutes.

Force Authentication by Token

When SSO is enabled, but if SSO failed, the user could still fall-back to logging in using their username/password. The setting turns off the ability for a user to do so.

This setting should be turned on for integrations that does not expose a way to validate by username/password.

Remember Me

This setting controls the cookie session, which is set to expire in 365 days. When this setting is enabled, it allows offline access (if granted via a DRM setting) and should reduce the number of log in sessions required.

Web Viewer SSO Request

The best way to take full advantage of Vitrium's Web Viewer SSO functionality with External Service is to generate an arbitrary token and include it in the request to Vitrium's content, either in a cookie or as part of the query string, as described above. This token can be any string. Vitrium will parse this string out of the request and supply it in the payload of the call to your External Service.

The best way to illustrate this powerful functionality is with an example. Let's say that in order to authorize a user's access to content, you need to know the user's ID, an order number in your system, and a subscription expiration date. You would generate a token containing this information. For example, you could take a string "user:123,order:xyz,expiry:20201231". You could quite easily encrypt this using secret key and vector known only to you, then convert this hash to a Base64 URL-friendly string.

NOTE: Standard Base64 encoded string may contain '+', '/', and '=' characters, which causes issue in URLs and is best avoided. Using Standard Base64 with URL may cause intermittent failures that could be difficult to track down.

The details of generating this token are entirely up to you. This value would then be provided in the Web Viewer request to protected content, either in the cookie or query string. The name of the cookie or query string parameter needs to match exactly the Query Parameters or Cookie Names SSO Setting.

When this request arrives at the Vitrium Web Viewer, the request is examined for query string or cookie parameters matching the configured settings. If a match is found, the token value is parsed out and sent to the Vitrium Auth component, along with other values and tokens. When the request arrives here, it in turn is inspected for a field whose name match the Web Viewer SSO Token configured under Account Settings. It is therefore important to configure these two values – Account Settings > Web Viewer SSO Token and SSO Settings > Query Parameters/Cookie Names – to be the same value (technically, since the SSO Settings are lists, they need to *contain* the WV SSO Token from Account Settings). If a *case-sensitive* match is found, the value is then passed to your External Service in the Token field.

Let's continue with our example where you wanted to pass the value "user:123,order:xyz,expiry:20201231" as a SSO token to our External Service. You would then create a hash or somehow encrypt this string to arrive at a token that we can safely pass as part of a query string. You would also need to configure your Query Parameters and Web Viewer SSO Token to an arbitrary but matching value, for example "ssoToken". You might then create a URL with this query parameter similar to this:

https://view.protectedpdf.com/contentAlias123?ssoToken=2QwMEdNZC9LS1JUc1JjNHZwNFRuRkpzUG9PYmdQLw%3D%3D

You could make this URL a link in your content management portal or distribute it to your customers via email; regardless of the method, the user would click a link to view this protected content. The request would be handled by Vitrium's Web Viewer and Auth components, and a request would arrive at your External Service's `/authenticate` endpoint with a payload similar to this one:

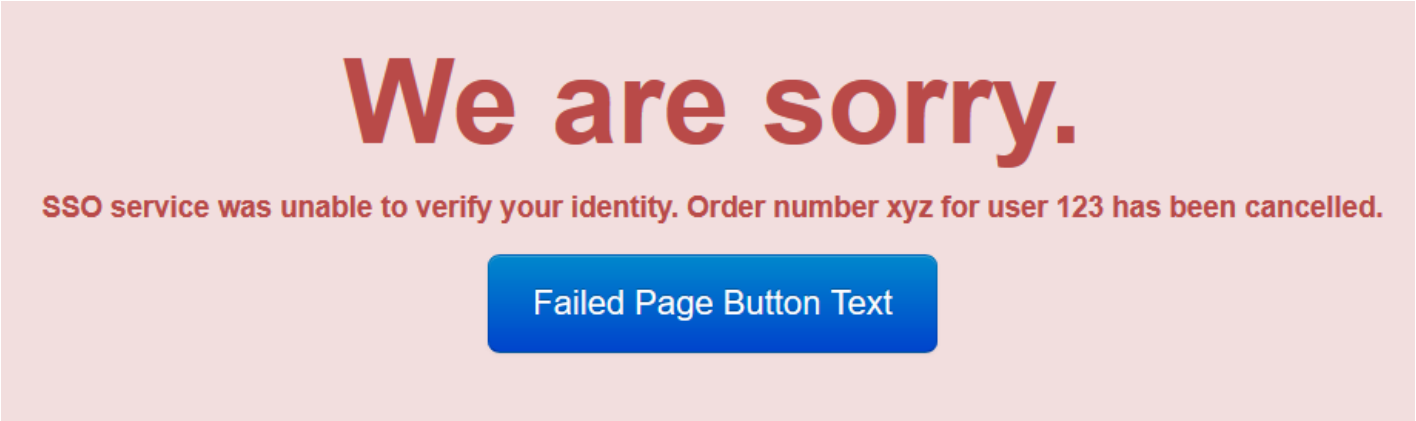

```
{"Username":null,"Password":null,"HashingKey":null,"HashingVersion":null,"CaseSensitiveP
assword":false,"Token":"2QwMEdNZC9LS1JUc1JjNHZwNFRuRkpzUG9PYmdQLw==","Type":"WebViewerSs
o","Document":{"FolderPath":["843200d7-77c5-4fac-abaf-
da8311783ff9"],"DocumentId":"10ca98be-400f-48d7-98bc-
90e6a1f849f2","VersionId":"3f12dd64-5907-4e34-9170-75edccc28e68","DocCode":"5000-65F3-
108F33-
0010AC9D","Metadata":{"ContentType":"std","FileName":"sampleSmall1","VersionName":null,"
UserSpecificWatermarkTemplates":[{"Id":"f7dff30b-da44-44b3-9d63-
96cbfa6cba8d","Name":"default","TextTemplate":"Licensed to
_userName_"}]},"ExternalKey":null,"Status":{"IsActive":true,"IsMostRecentVersion":true,"
IsMostRecentVersionActive":true}},"UserClient":{"AppName":"Firefox","AppVersion":"74.0",
"Platform":null,"OperatingSystem":"Windows 10 .","DeviceName":"Other","DeviceId":"WV-
```

7a53f1d0-f90b-4801-9420-
09af9041b4fb","HasOfflineAccess":false,"InjectVersion":null,"IpAddress":"248.158.108.69
","Language":"en","OutOfBrowser":false,"ServerUrl":"https://view.vitrium.com/contentAlia
s123?ssoToken=2QwMEdNZC9LS1JUc1JjNHZwNFRuRkpzUG9PYmdQLw%3D%3D"}}

You "would inspect the Type field and see that it is “WebViewerSso”, which means you need to handle the value of the Token field. Because you (and only you) know how the token was generated, you know how to unwrap, unhash or decrypt it to arrive at the information contained within it. In this case, you would unpack the token to the original value of “user:123,order:xyz,expiry:20201231”. You would check to see if the user ID and order number are valid and active in your system and return the appropriate response with HTTP Status 200 (OK). For example, you might determine that the user ID is invalid or that the order has been cancelled and return a response with a payload like this:

```
{
  "Succeed": false,
  "Message": "Order number xyz for user 123 has been cancelled."
}
```

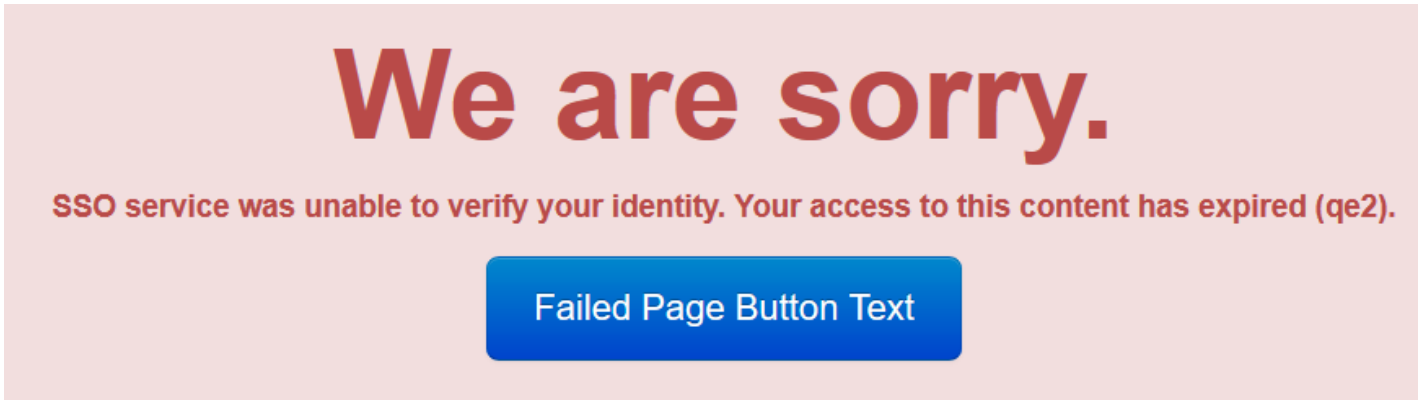
Provided that the SSO Settings are configured so that the end user is presented with a SSO Fail Page rather than being redirected immediately, they would see a page like this:



Or you might determine that the user is active, and order is valid, but that the subscription expiration date is actually 12/31/2019 and return a response payload like this:

```
{
  "Succeed": true,
  "Policy": {"Expiry": "2019-12-31"},
  "UserId": "123",
  "Username": "user123"
}
```

This, of course, would also result in an SSO failure because access to the protected content has expired, and the user would see something like this:



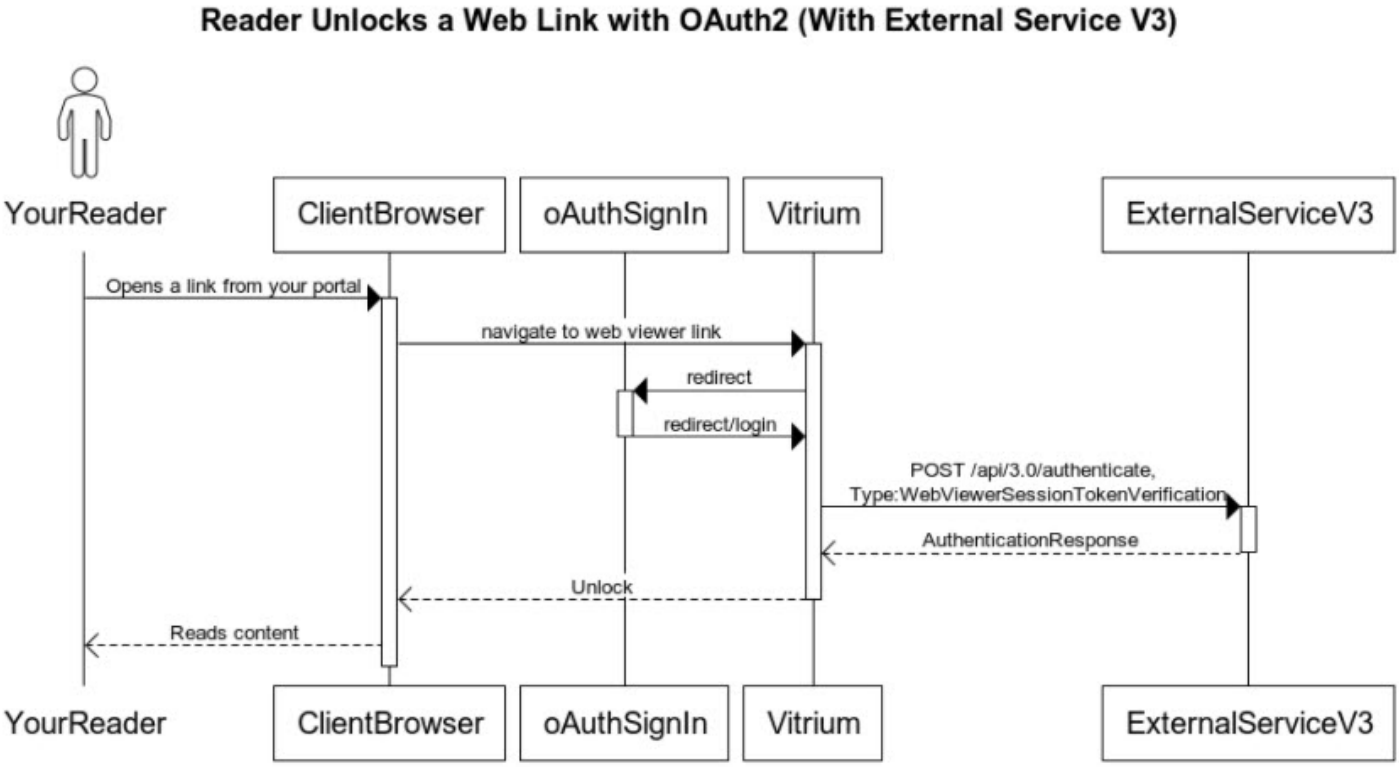
Only if you verify all the data in your system and return a valid access policy would the user be able to access the protected content.

One final note: Once a user is granted access to protected content at a particular Web Viewer URL, Vitrium creates a session cookie and uses it next time that URL is accessed. For example, if you grant access to a user through SSO to the content at the URL we've been using in this example, then a subsequent request to the same URL (https://view.protectedpdf.com/contentAlias123?ssoToken=2QwMEdNZC9LS1JUc1JjNHZwNFRuRkpzUG9PYmdQLw%3D%3D) would result in a WebViewerSessionTokenVerification request to your External Service, similar to this:

{"Username":"user123","Password":null,"HashingKey":null,"HashingVersion":null,"CaseSensitivePassword":false,"Token":null,"Type":"WebViewerSessionTokenVerification","Document":{"FolderPath":["843200d7-77c5-4fac-abaf-da8311783ff9"],"DocumentId":"10ca98be-400f-48d7-98bc-90e6a1f849f2","VersionId":"3f12dd64-5907-4e34-9170-75edccc28e68","DocCode":"5000-65F3-108F33-0010AC9D","Metadata":{"ContentType":"std","FileName":"sampleSmall1","VersionName":null,"UserSpecificWatermarkTemplates":[{"Id":"f7dff30b-da44-44b3-9d63-96cbfa6cba8d","Name":"default","TextTemplate":"Licensed to _userName_"}]},"ExternalKey":null,"Status":{"IsActive":true,"IsMostRecentVersion":true,"IsMostRecentVersionActive":true}},"UserClient":{"AppName":"Firefox","AppVersion":"74.0","Platform":null,"OperatingSystem":"Windows 10 .","DeviceName":"Other","DeviceId":"WV-7a53f1d0-f90b-4801-9420-09af9041b4fb","HasOfflineAccess":false,"InjectVersion":null,"IpAddress":"248.158.108.69","Language":"en","OutOfBrowser":false,"ServerUrl":"https://view.vitrium.com/contentAlias123"}}

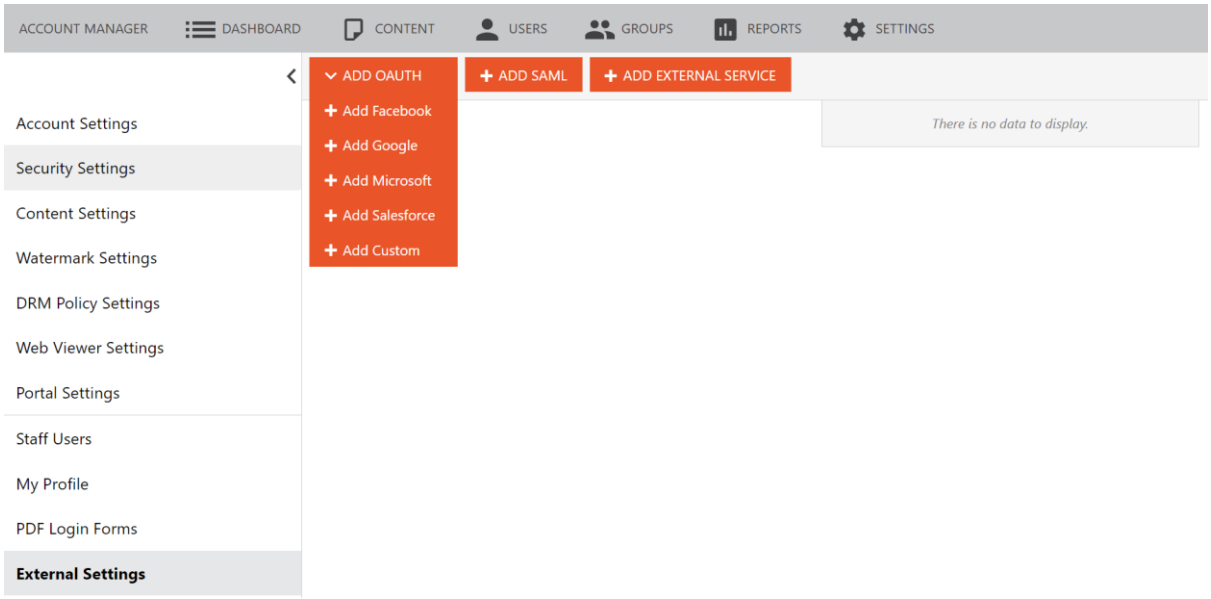
This is despite the fact that the URL contains a token and all SSO Settings are properly configured: if a valid Web Viewer session cookie exists for a particular content alias/ID, Vitrium will by default attempt to use it. You can deal with this by handling the WebViewerSessionTokenVerification request appropriately – it will contain the username of the user which you know was previously authenticated by SSO and successfully accessed the content. However, if your use case requires that the SSO token is always parsed and sent to your External Service, you can suppress this default functionality in Vitrium Web Viewer by checking Force Authentication by Token in your SSO Settings. If that setting is checked, Vitrium will always first attempt to parse the SSO token as described in this example and send it in the payload or a WebViewerSso request.

Web Viewer by OAuth Workflow

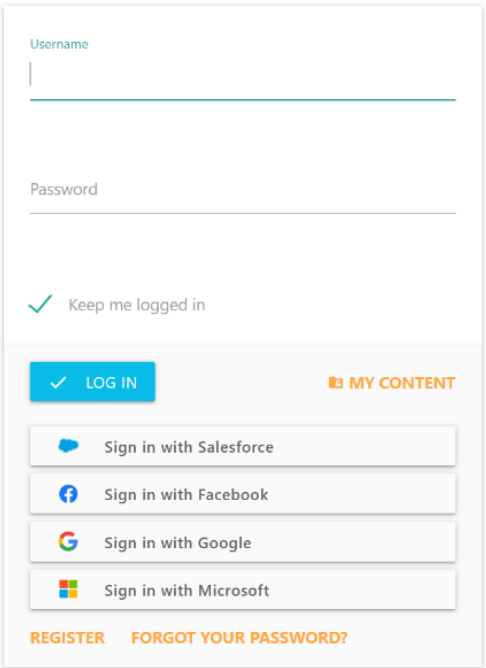


Vitrium External Settings Configuration

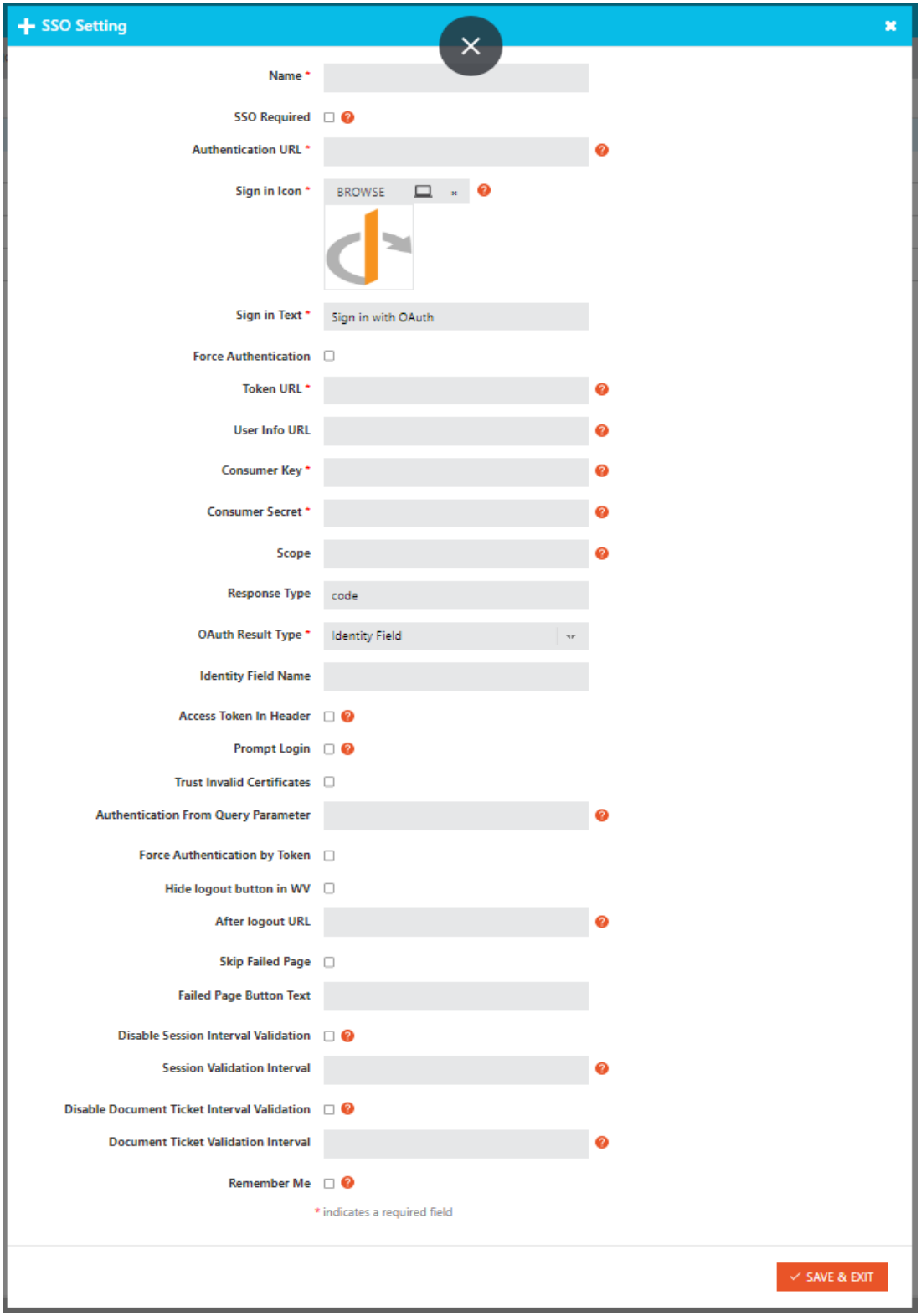
In order to take advantage of Web Viewer SSO by OAuth functionality you first need to configure some settings in the Vitrium Admin UI:



You can configure multiple OAuth configurations, with 4 templates provided to use - Google, Microsoft, Facebook and Salesforce. Once an OAuth and/or SAML configuration is completed and enabled, a button will appear on the login form - both Web Viewer and Portal - allowing you to either enter Vitrium credentials or log in with one of the configured external options.



OAuth settings:



Name

Account SSO setting configuration unique name.

SSO Required/Authentication Url

Authentication Url is required when SSO Type is set to OAuth. Set Authentication Url to your OAuth sign in page.

If SSO Required is enabled [Recommended]

- Non-logged in end users that navigated to a web viewer Url are redirected to the Authentication Url
- End users that tried to login, but failed are redirected back to the Authentication Url to try again
- End users that are logged out are redirected to the Authentication Url

If SSO Required is disabled [Review with Vitrium first]

- Non-logged in end users that navigated to a web viewer Url are redirected to the Authentication Url
- End users that tried to login and failed are redirected to the Vitrium Web Viewer login form to allow them to manually login using their credentials. The External Service needs to be able to Authenticate with username/password.
- End users that are logged out are redirected to the Vitrium Web viewer login form

Sign in Icon

Icon that will be shown on the login form for this configuration.

Sign in Text

Text that will be shown on the login form for this configuration.

Token Url

This is the endpoint which issues an OAuth 2.0 access token.

User Info Url

This is the OpenID Connect UserInfo endpoint, used by Vitrium to obtain data which can be used to retrieve an enduser/reader. This value is logged in Vitrium as the UserName (email address) for activity logs and analytical reports. It's used to retrieve the "reader" to authorize the user to the document. The authorization is done in Vitrium using the DRM/permission settings. Refer to the parameter "IdentityFieldName" that defines the expected return element.

Hide Logout button in WV (Web Viewer)

If you have a business case to disallow users from having access to the log out button in the Web Viewer.

After Logout URL

If user clicks the Web Viewer logout button, they will be redirected to this URL rather than the default Vitrium Web Viewer login page/form.

Skip Failed Page

If set to true (1) and SSO Authentication fails, the end user is presented with a Failed page explaining that SSO Authentication failed. The page includes a button (see FailedPageButtonText) redirecting the user to either your Authentication URL or the Vitrium Web Viewer login screen, depending on the value of the IsRequired setting. If this

SkipFailedPage setting is false (0), this intermediate page is not displayed, and the end user is instead redirected immediately to either your Authentication URL or Vitrium Web Viewer login screen.

Failed Page Button Text

This setting is only relevant if SkipFailedPage is set to false (0) and can be used to customize the text of the button redirecting the user as explained above.

Disable Session Interval Validation

Disables session token expiration. It ensures that SSO backend service is contact only once per session lifetime.

Session Validation Interval

Number of minutes before session token expires. Default is 90 minutes. When session token expires, we talk to backend to validation session again.

Disable Document Ticket Interval Validation

Disables re-verifying user access to content

Force Authentication by Token

Interval in minutes for re-verifying user access to content. Default is 5 minutes. Increasing this interval will ease load on your external service but impacts its responsiveness and delay revoking contact access.

RememberMe

if enabled (1) a web viewer session is set in a cookie with expiry of 1 year from now FOR THE CURRENT CONTENT. If not enabled (0) then they will need to authenticate each time. RememberMe needs to be enabled for “Save to cache” to work.

ConsumerKey

Consumer Key (AKA Client ID). Must be obtained from the OAuth2 provider. The value is passed to the AuthenticationUrl and TokenUrl.

ConsumerSecret

Consumer Secret (AKA Client Secret). Must be obtained from the OAuth2 provider. The value is passed to the TokenUrl.

Scope

OAuth2 Access Token Scope. Ie. Openid

Response Type

This indicates what values are returned by the OAuth authorize/ endpoint. The allowed values are "code", "token" (access token), and "id_token", but the values can be combined. Vitrium currently only uses the "code", which is our default value. So technically we can allow "code", "code token", "code id_token", or "code id_token token", but in all cases we only use code and ignore any tokens.

OAuth Result Type

If set to "IdentityField", then we look for the user element in "IdentityFieldName".

Identity Field Name

"Enabled if OAuth Result Type = ""Identify Field"". We look for this data element in the UserInfoUrl response. This becomes the username that is then logged in Vitrium. Ie. ""email"" or ""preferred_username""

Access Token In Header

Used by some OAuth2 environments. When enabled, OAuth2 authorization access token will be passed in the header. If disabled, the URL parameter will be used.

Prompt Login

If set (1), we append "&auth_type=reauthenticate&prompt=login" to the AuthenticationUrl.

Authentication From Query Parameter

Url query parameter name that is going to contain the document Url where you are redirected from. Eg. https://authentication.service.com/?from=https://view.vitrium.com/abcde

Trust Invalid Certificates

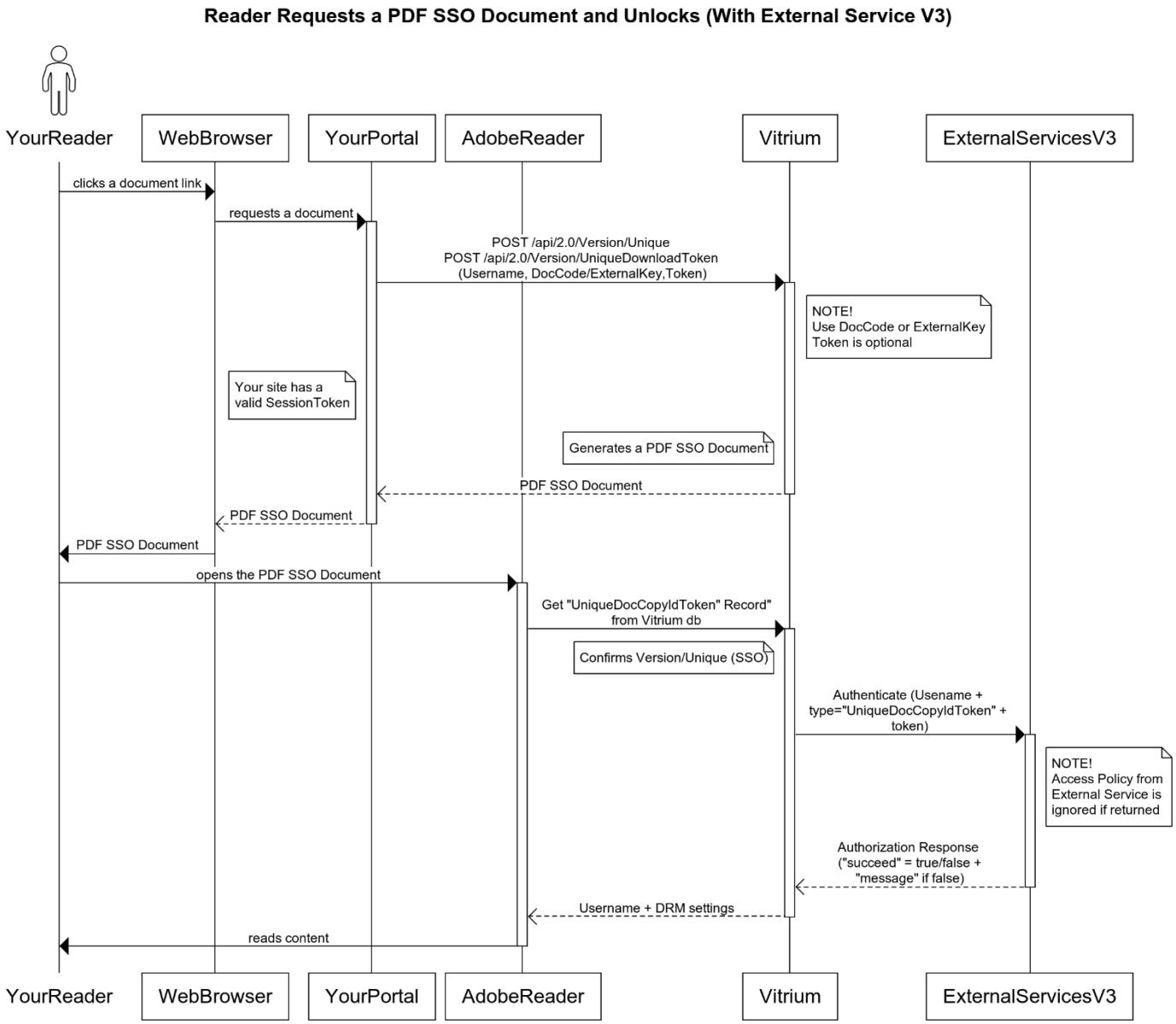
Checking this checkbox instructs Vitrium to ignore invalid SSL certificate during OAuth sign-in. This should only be used on development systems.

PDF by Version Unique

Vitrium SSO PDF works by embedding a unique code inside each PDF. The unique code is then associated a User.

This workflow consists of two steps, Generating the PDF and Unlocking the PDF.

PDF Workflow



Generating the PDF

When a user clicks to access content in your portal, your portal can generate an SSO PDF by calling the one of the Vitrium API below with the Document Code (or ExternalKey) and the Username.

- POST version/uniquedownloadtoken

- POST version/unique
- POST version/merge

If using the version/unique API, it will respond with the PDF document with the unique code embedded into it. Your portal needs to retrieve and send it to that User.

If using the version/uniquedownloadtoken API, it will respond with tokenized URLs that you will deliver to the User and they download the protected pdf directly.

In all the "unique" API's you can include an optional "token" that will be sent to the external service along with the username when the user attempts to unlock the file they receive.

In the "unique" API's, you can reference a content stored in Vitrium System using the DocCode or your ExternalKey value that you would have earlier added to the content record.

Unlocking the PDF

When a user opens their SSO PDF, their unique code is sent to the server where it determines the Username that owns that PDF. The server validates the unique code against the Vitrium database and, if it exists, then validates the DRM setting and allows the unlock to occur or is denied.

The server will then send a request to the external service using a Type of "UniqueDocCopyIdToken" and send the username and the optional token if it was included in the initial version\unique API call.

Revoking the PDF

To revoke user access to a previously provided version/unique Protected PDF, you will need to have the previously generated unique id. Use this API to look it up:

- GET GetUniqueDocumentsByDocCodeAndUsername

And then use this API to revoke their permission to the content:

- DELETE version/unique

There is no API to change or PUT new values so you must use the GET, DELETE, and a new POST with the original version id to change the previously assigned DRM settings … to reduce or increase the number of devices as an example, or to extend an expiry date, etc.

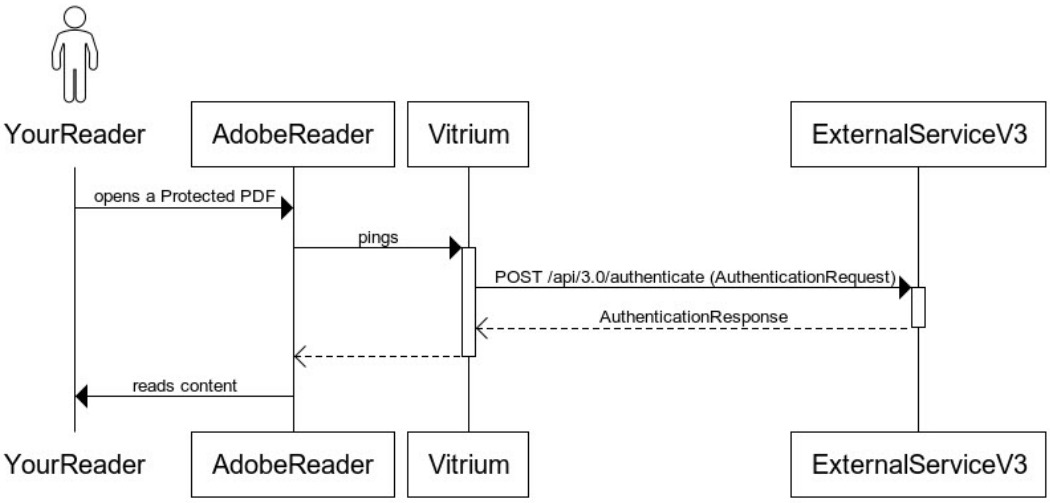
For more information on calling Vitrium's API, please reach out to our Support team.

"SSO Lite" Unlock

When a user opens the first PDF, the document calls Vitrium where our software associates the DeviceId used with the user. Next, when the user opens the same or another PDF, the Vitrium server sees the DeviceId and looks up the user. The server then issues an Authenticate call with its Type field set to **"SsoLiteToken"**.

The Token field in the request holds the Id of the user who is unlocking a document through SSO Lite. You should verify that the User for the Id exists and is allowed to unlock content.

Reader Unlocks a PDF SSO Lite (With External Service V3)

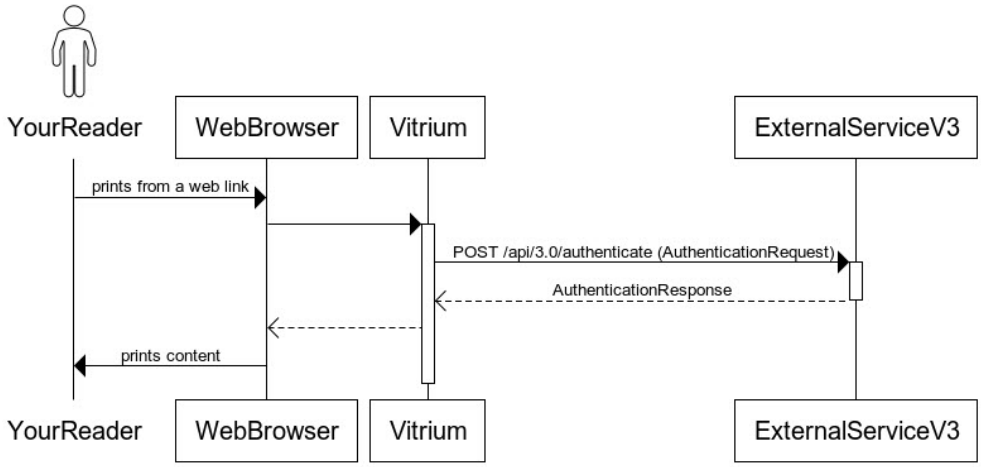


Vitrium could also perform SSO Lite using IP Address instead of DeviceId. This setting is available in the Settings tab.

Print Metering

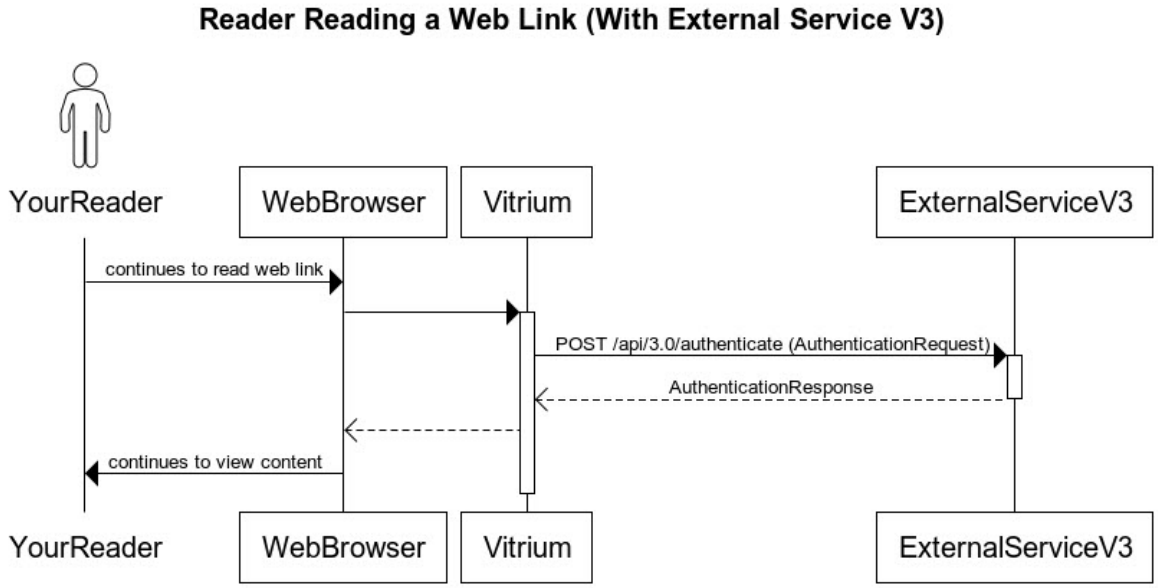
When a user is printing a document in the Web Viewer, the server issues an Authenticate call with Type **"PrintMeteringUsernameToken"**. The Username field in the request holds the username of the User. You should verify that the User for the username exists and is allowed to unlock content.

Reader Prints from a Web Link (With External Service V3)



Web Viewer Session Token Verification

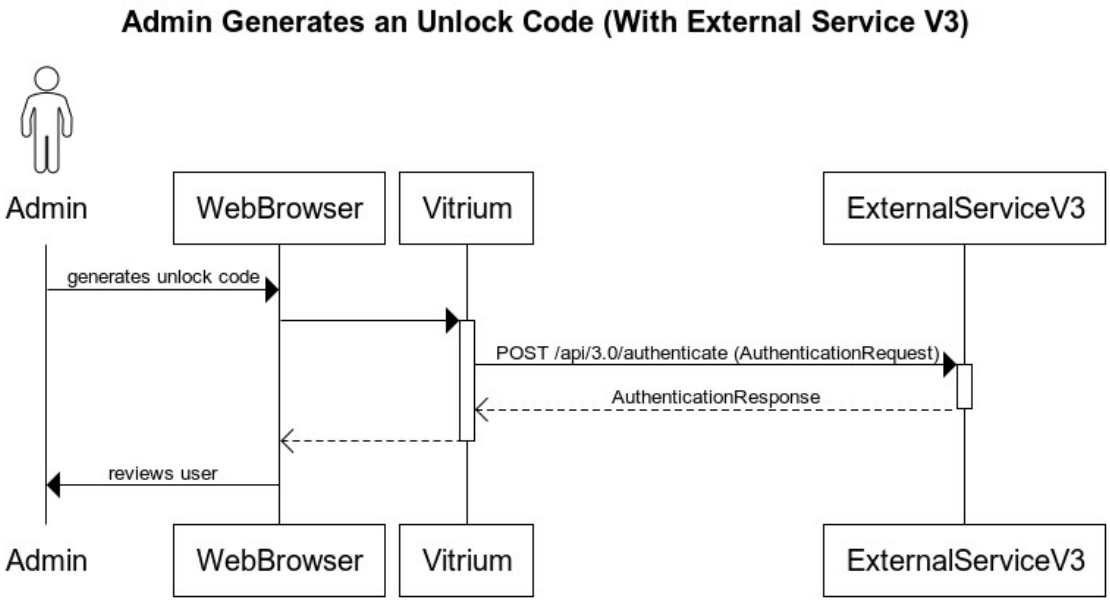
When a user keeps a document open in the Web Viewer, the server periodically issues an Authenticate call with Type **"WebViewerSessionTokenVerification"**. The Username field in the request holds the username of the User. You should verify that the User for the username exists and is allowed to unlock content.



Remote Unlock Token

When an administrator generates a remote unlock code for a user, the server issues an Authenticate call with Type **"PhoneUnlockToken"**.

In the request, the Username field holds the username of the user and the Document field holds the metadata for the content. You should verify that the User for the username exists and is allowed to unlock the specified content.



Secondary External Service Endpoints

User Listing

To clear past user usage, you would need to be able to access your external users in the User tab of the Vitrium software. To accomplish that, the “readers” (plural) endpoint is required in the external service and the response from that endpoint will then display users.

IMPORTANT!

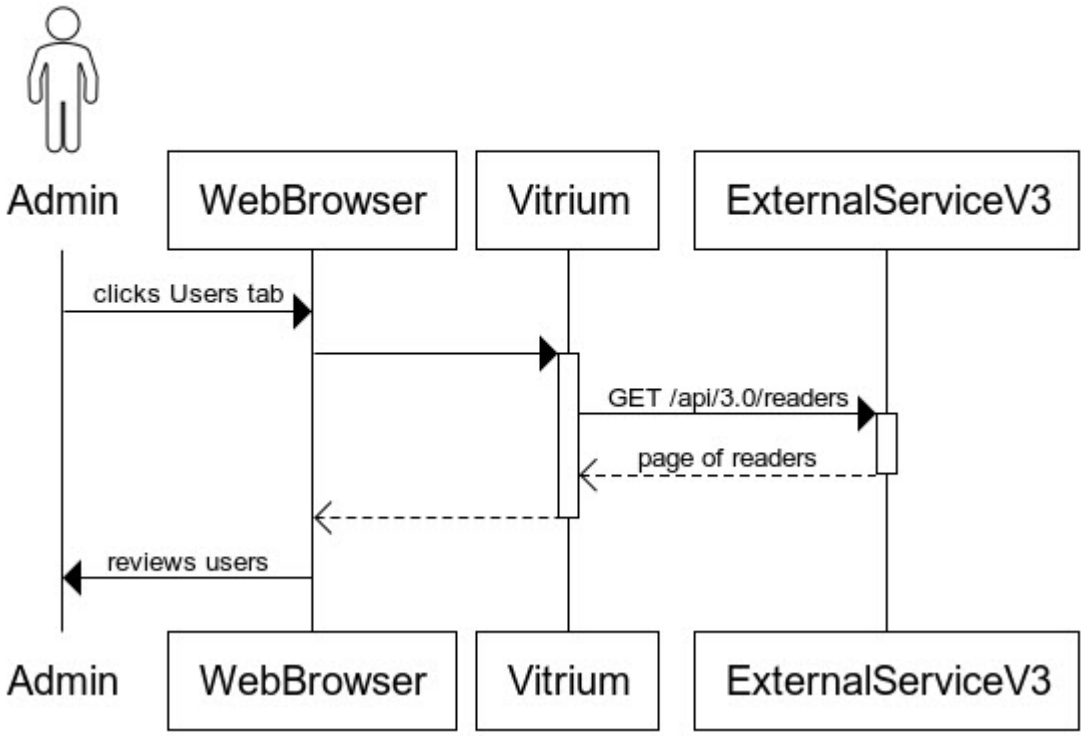
It's important that you implement the ***Readers*** endpoint in your external service as without it your support staff will not be able to manage end-user issues such as device or print limits and staff will not be able to reset those DRM counters. The ***Readers*** endpoint is used to populate the users/readers window in the Vitrium Admin UI under the Users tab. It's provides search functionality and the ability to traverse list of users. Most importantly, it provides needed functionality such as reviewing a User’s unlock history as well as clearing past usage. If the Readers endpoint is not implemented, this much needed functionality will not be available. Refer to the ***Readers*** API specification on page 24 for details

NOTE! In September 2022 (build 10.8), a new feature setting was added to the External Settings on the Account Settings windows in Vitrium software that would populate the User tab based on users that had activity (past unlocks) of content. This would eliminate the need to build this endpoint in your external service. It can be enabled by DISABLING the “Show External Readers” checkbox. The build-in feature will display user created dates representing the first time they unlocked a content record. If the checkbox is enabled, then the system will ping your external service and you would then be expected to return a similar list of users.

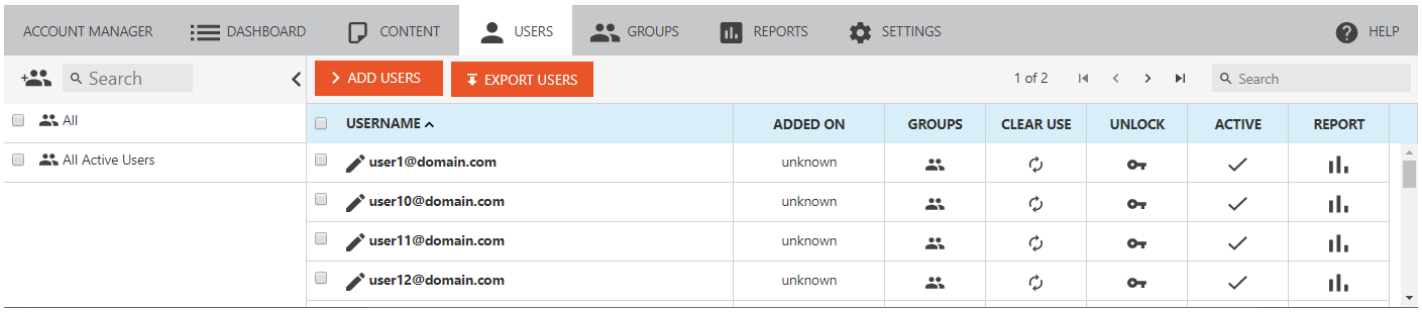
The following API is called:

API Call	Purpose
GET /api/3.0/readers	Shows users in Vitrium’s administrative user interface (in the Users tab)

Admin Clicks on the Users Tab (With External Service V3)



The results are then shown in the Users tab and the Clear Use buttons can then be accessed:

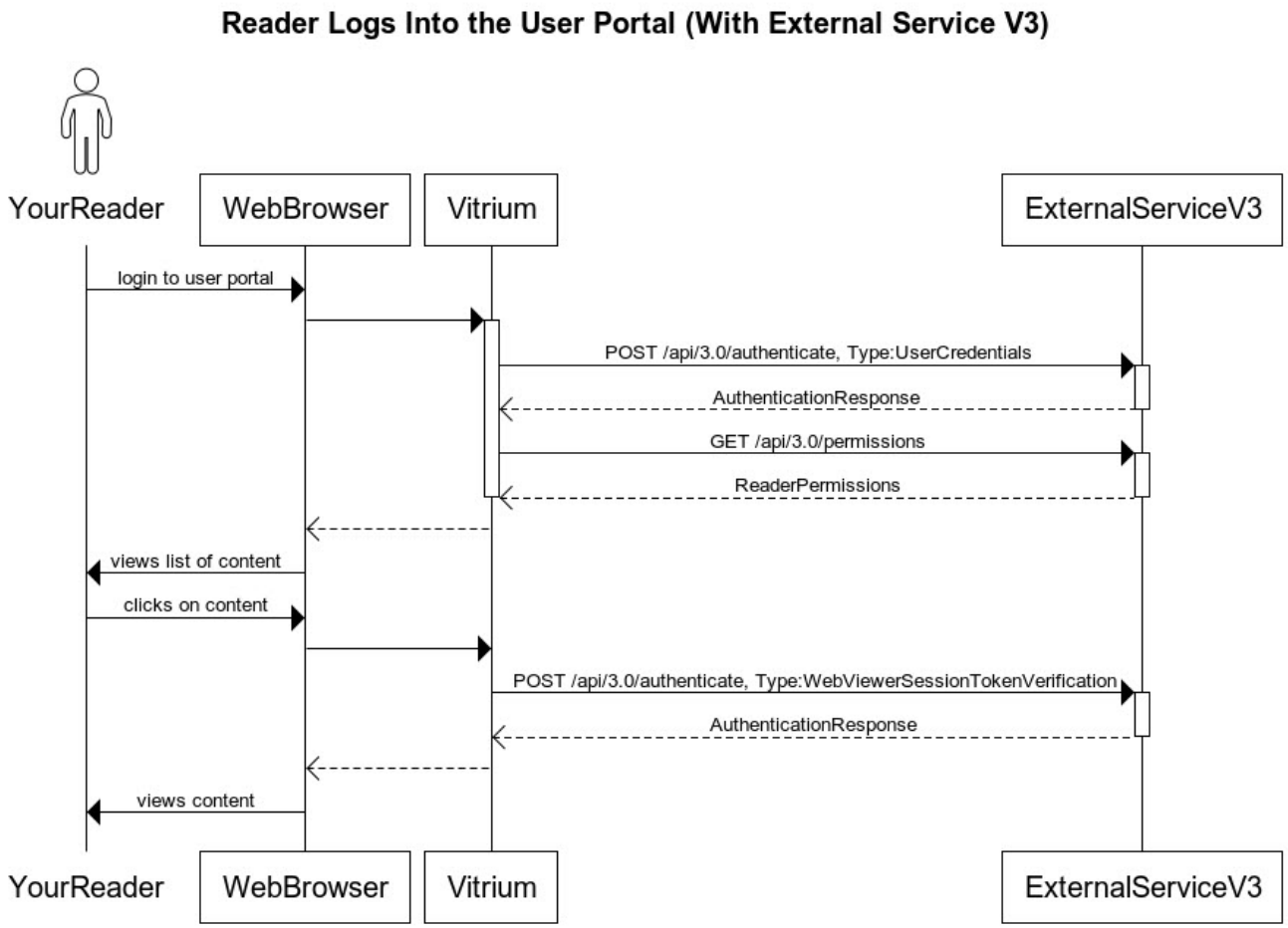


Vitrium User Portal

When a User visits the Vitrium User Portal, the following APIs are called:

	API Call	Purpose
1	GET /api/3.0/authenticate Type: UserCredentials	Checks that the username/password for the User is correct and has access to the Content.
2	GET /api/3.0/permissions	Get a list of permissions for the User for the Web Portal
3	GET /api/3.0/authenticate Type: WebViewerSessionTokenVerification	Checks that the User for the username is valid and active.

The authenticate API must be implemented and tested before tackling the Portal workflow. You will then need to implement the permissions endpoint to enable Vitrium User Portal related functionality. The User Portal uses the permissions API to determine what Content to show the User in the User Portal. Refer to the following sequence diagram.

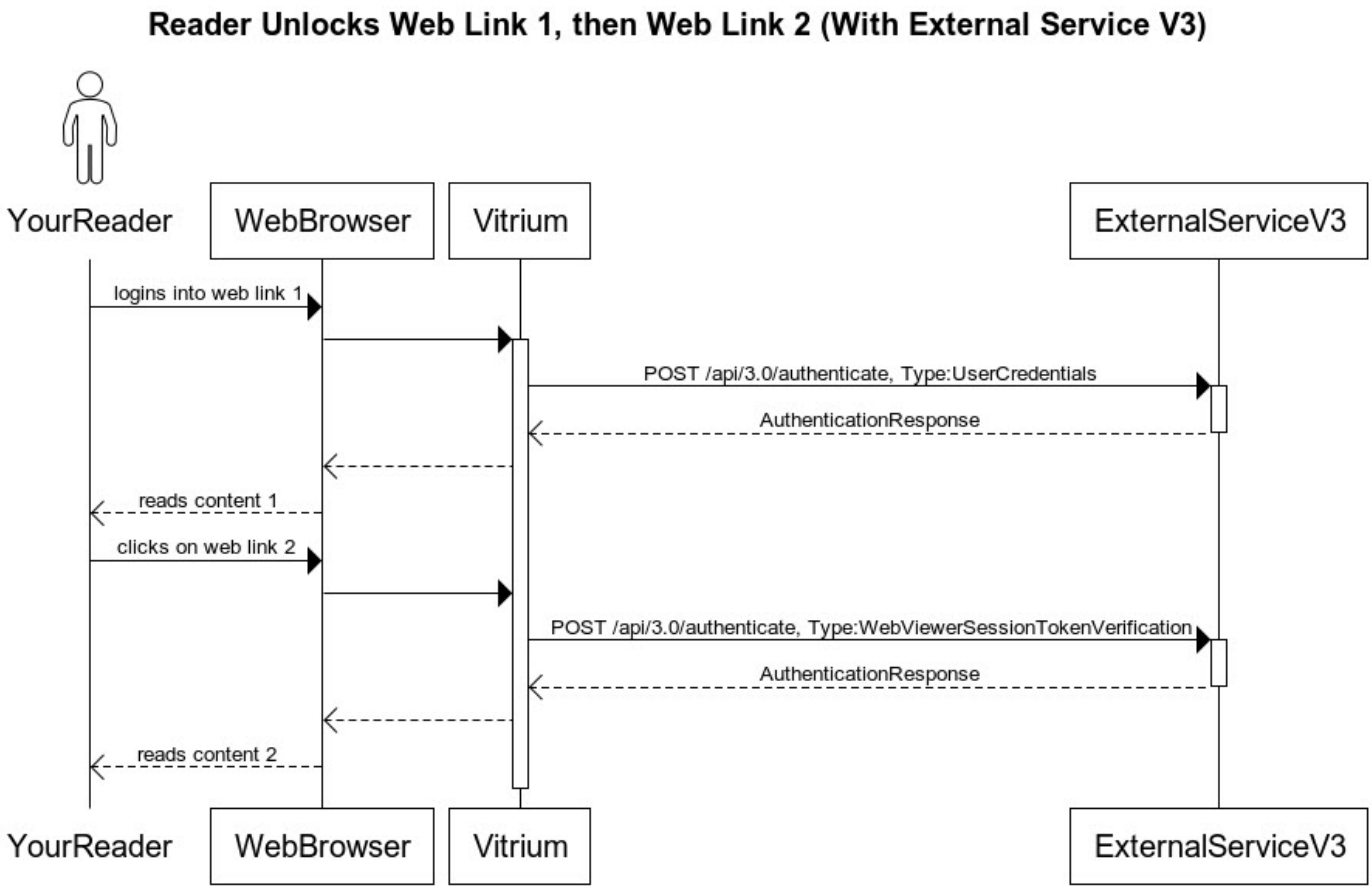


Web Viewer Session Verification Overview

In this first example, a user would initially login manually to the Web Viewer (browser) using a username and password and when they subsequently open the same or other Web Viewer links, the cookie is used to send re-authentication codes.

- If "Force Authentication by Token" is unchecked, then the subsequent WebViewerSessionTokenVerification request will contain "Username": {originally authenticated user's username}
- If "Force Authentication by Token" is checked, then the subsequent WebViewerSessionTokenVerification request will contain "Token": {original token as supplied in the query string}

In the following diagram, the initial login (clicks on web link 1) is manual - user enters credentials - and in the second request (clicks on web link 2), Username will always be sent. The other scenario would be an identical looking diagram, where the initial type would be "WebViewerSso" instead of "UserCredentials", and then the bullet points would apply to the second WebViewerSessionTokenVerification request



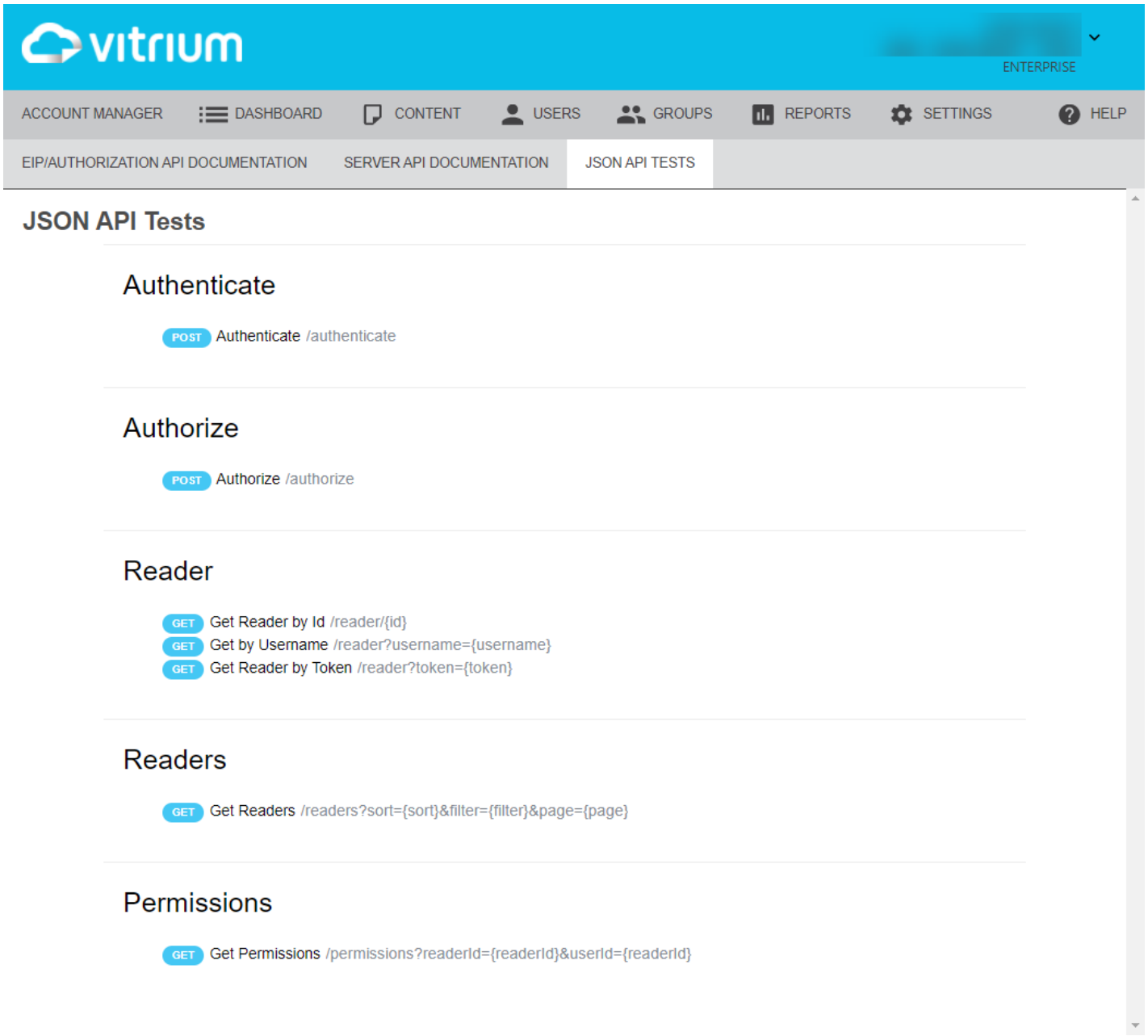
In this next example, a user would initially login to the Web Viewer (browser) using a username and password and when they subsequently open the same or other Web Viewer links, the cookie is used to send re-authentication codes.

- If "Force Authentication by Token" is unchecked, then the subsequent WebViewerSessionTokenVerification request will contain "Username": {originally authenticated user's username}
- If "Force Authentication by Token" is checked, then the subsequent WebViewerSessionTokenVerification request will contain "Token": {original token as supplied in the query string}

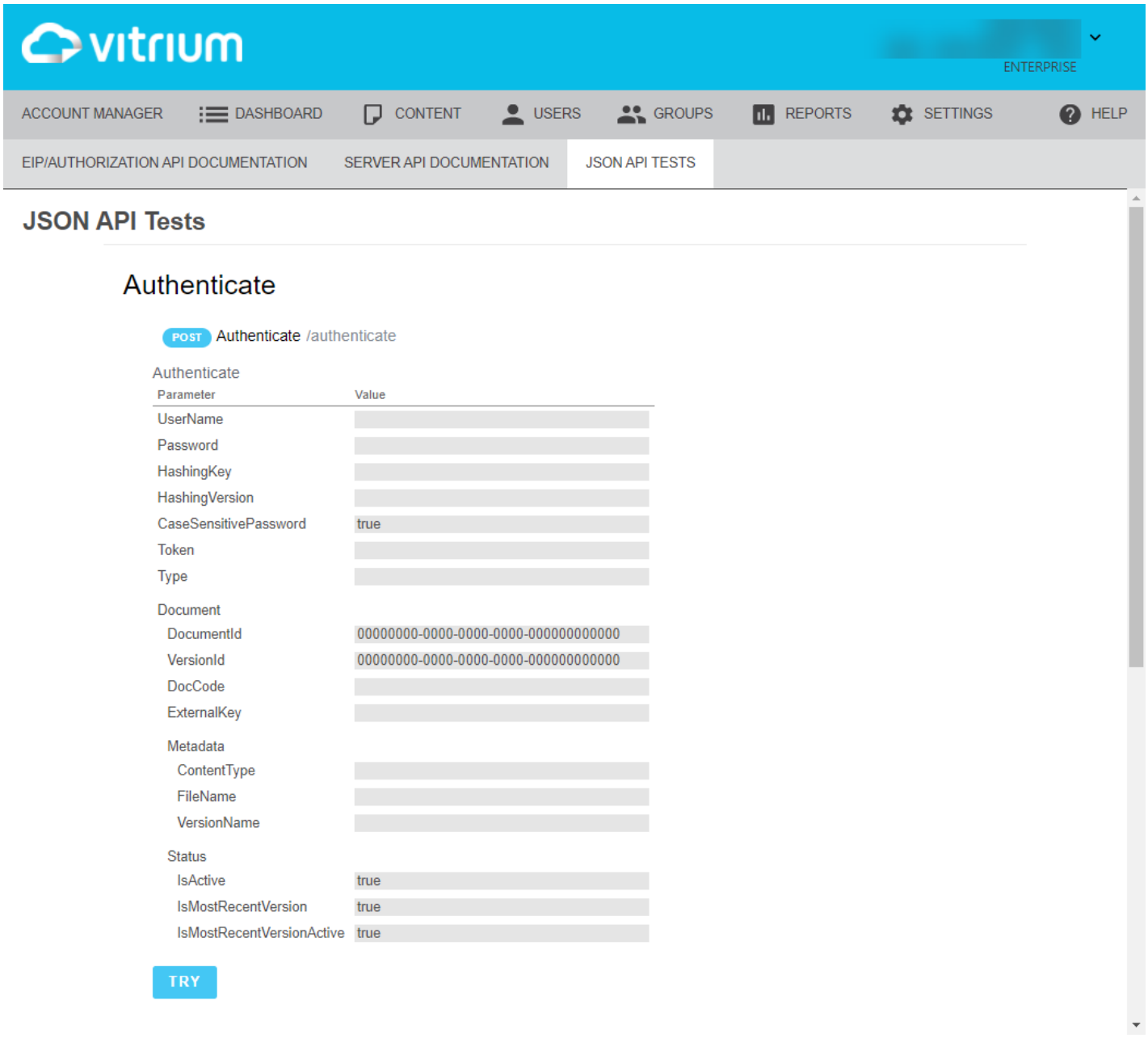
Testing Your External Service JSON APIs

NOTE: This updated V3 JSON test page is only available in Vitrium Security v9.x. To check/verify the version of Vitrium Security that you are using, while logged into the Admin UI change the browsers URL such that it opens the "/info" endpoint. For example, https://security.vitrium.com/info

You can access the JSON API Tests from the Help tab within your Vitrium Enterprise account.

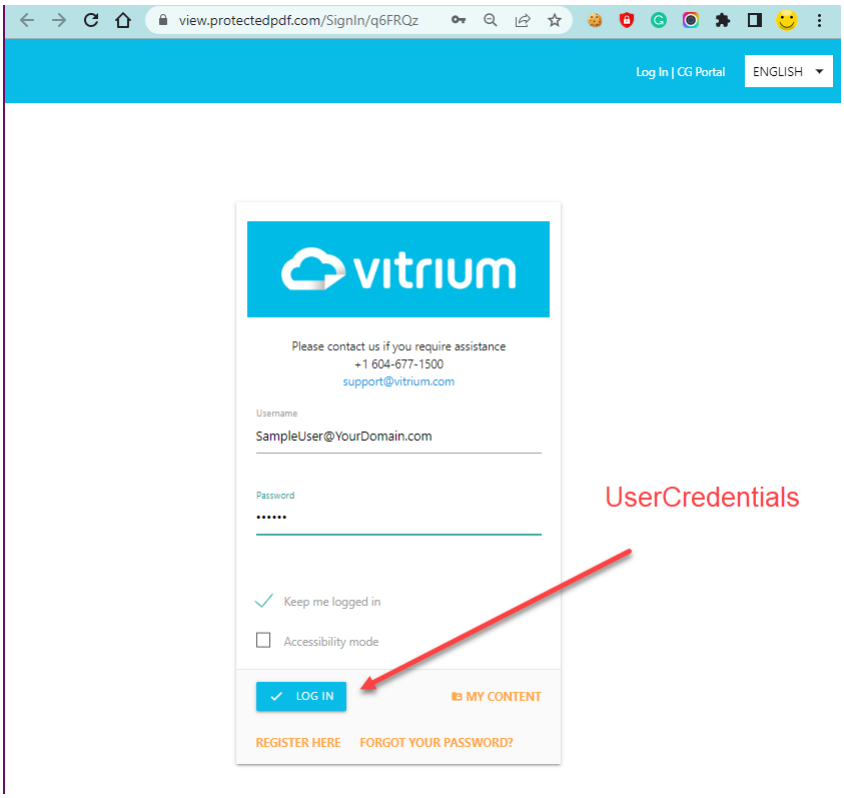


Click the POST button on any method to open the method input form. Fill in the key values that you would like to test and then click the TRY button at the bottom. A window will appear with the response from the external service.



External Service Authenticate "Type"

UserCredentials

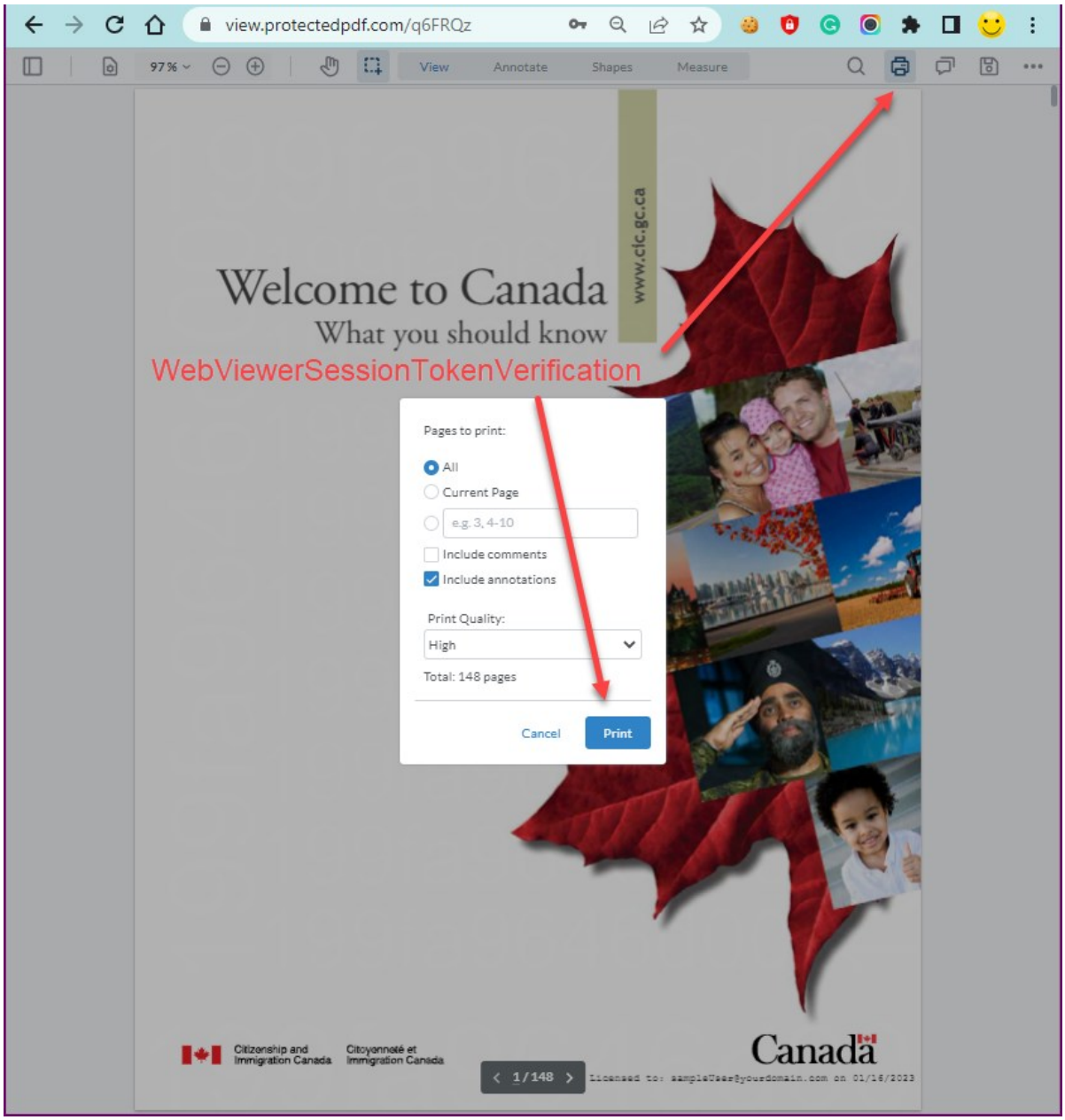


The "UserCredentials" request is made when a user is logging into the Web Viewer portal. This request will compare the login credentials entered into the portal, and if matched, will allow the user to log in.

When Type is "UserCredentials", the Authenticate Endpoint needs to handle four situations **in the following order**:

Username has a string, Password has a string, Document has a value	Verify that the username and password are correct. Verify that the reader has access to the document specified. This is not required if any form of SSO is used.
Username has a string, Password has a string, Document is null	Verify that the username and password are correct. This is not required if any form of SSO is used.
Username has a string, Password is null, Document has a value	Verify that the username is valid. Verify that the reader has access to the document specified.
Username is not null	Verify that the username is valid.

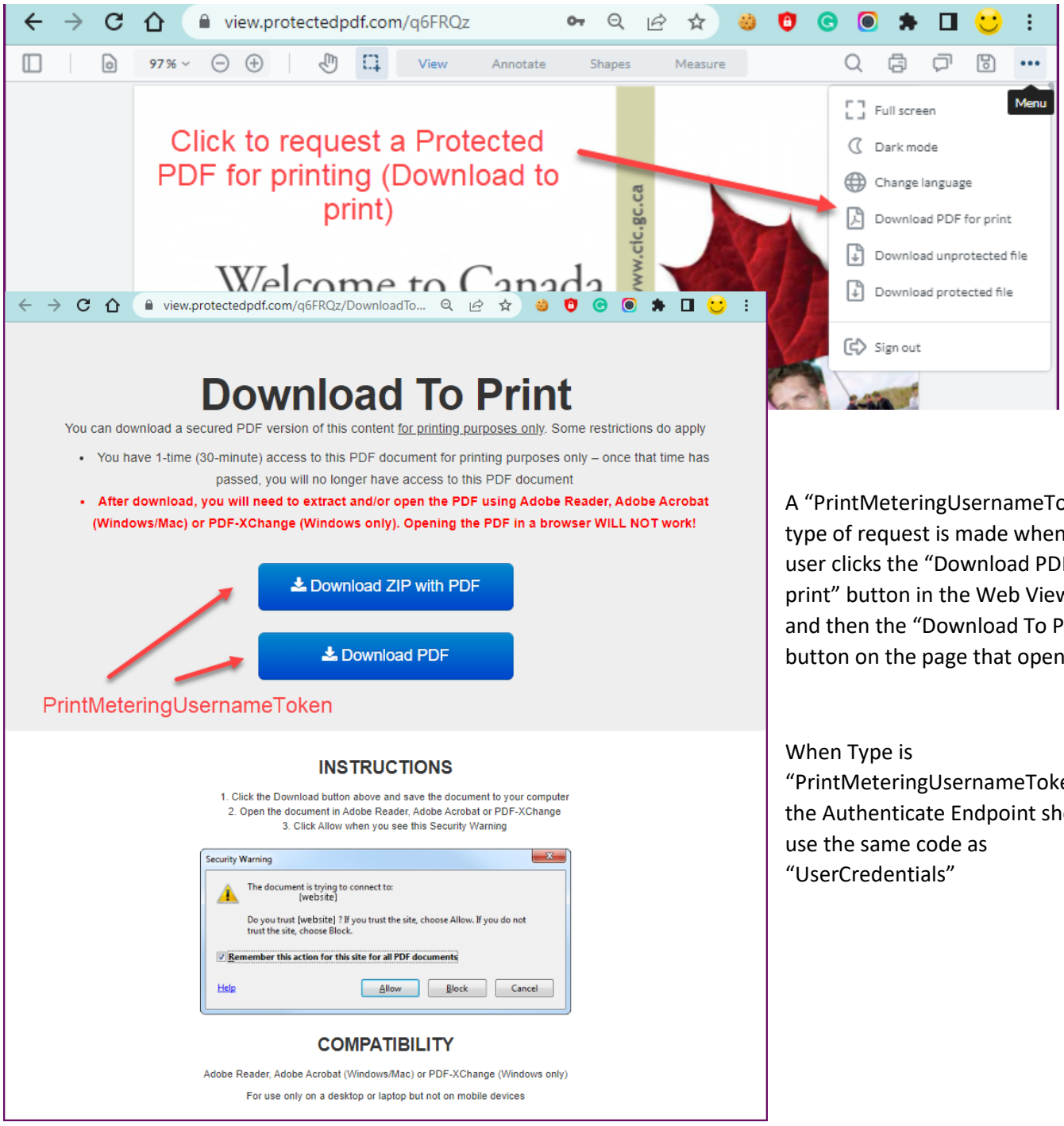
WebViewerSessionTokenVerification



"WebViewerSessionTokenVerification" type of request is made when the user clicks the "Print" button in the Web Viewer.

When Type is "WebViewerSessionTokenVerification", the Authenticate Endpoint should use the same code as "UserCredentials". By default, this will contain a Username unless your external service settings indicate to send the token "Force Authentication By Token".

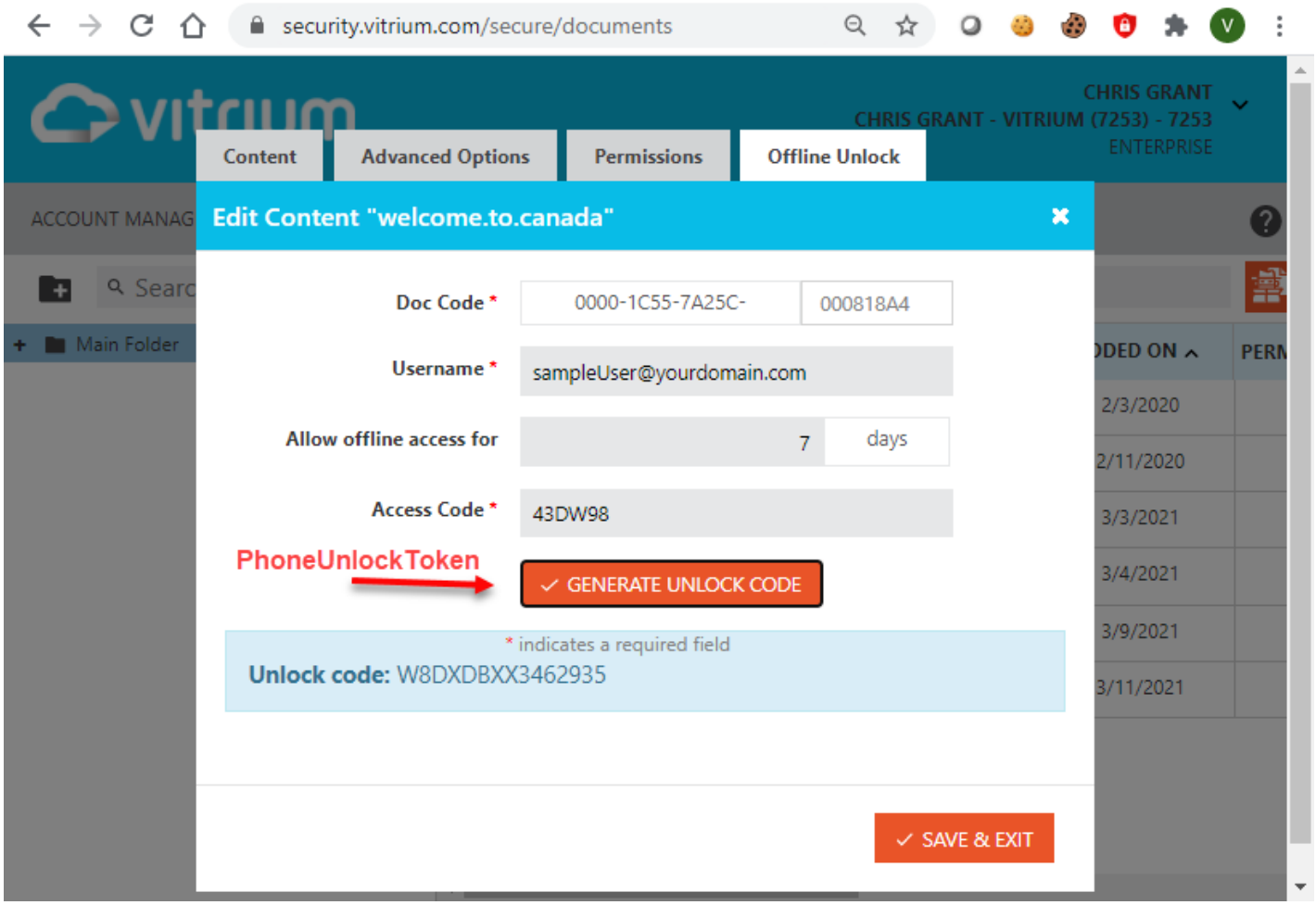
PrintMeteringUsernameToken



A "PrintMeteringUsernameToken" type of request is made when a user clicks the "Download PDF for print" button in the Web Viewer, and then the "Download To Print" button on the page that opens.

When Type is "PrintMeteringUsernameToken", the Authenticate Endpoint should use the same code as "UserCredentials"

PhoneUnlockToken

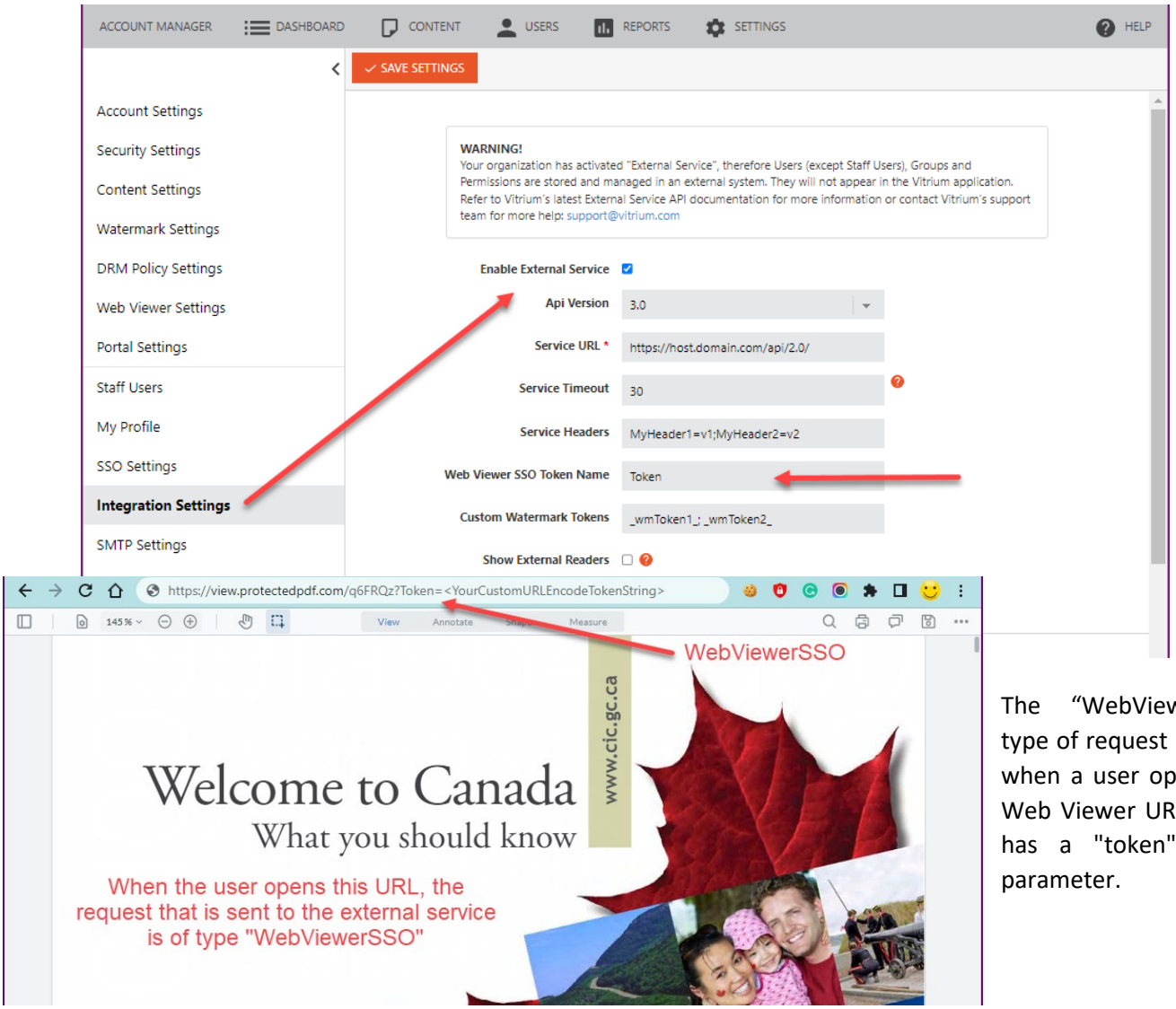


The “PhoneUnlockToken” is triggered when inside the Vitrium Admin UI, a user clicks a file, to edit its settings, then navigates to the “Offline Unlock” tab, fills out relevant information, and presses the “Generate Unlock Code” button.

If the (Doc Code + Username + Access Code) are not correct then the Unlock Code that is generated when the user clicks the Generate Unlock Code button will not be valid and the user won't be able to remotely unlock their protected PDF. The 3 variables are used to generate a hash code specific for the user that's requesting the unlock code for a given content record.

When Type is “PhoneUnlockToken”, the Authenticate Endpoint should use the same code as “UserCredentials”

WebViewerSso



The "WebViewerSSO" type of request is made when a user opens the Web Viewer URL which has a "token" query parameter.

When Type is "WebViewerSSO", the Authenticate Endpoint needs to handle two situations **in the following order:**

Token has a token representing the reader, Document has a value	Token contains a token representing the reader, provided by a portal via cookie or query parameter, so check whether the token is valid and that the reader represented by the token has access to the document specified. This is only required if cookie or query parameter are used
Token has a token representing the reader, Document is null	Token contains a token representing the reader, passed to the web viewer by your portal, provided by a portal via cookie or query parameter, so check whether the token is valid

WebViewerSessionTokenVerification

The "WebViewerSessionTokenVerification" type occurs every 5 minutes with the user is using a Web Viewer content. This transaction will occur when the user is sitting idle and even if the tab/browser is not currently in view.

When Type is “WebViewerSessionTokenVerification”, the Authenticate Endpoint should use the same code as “UserCredentials”

HashedUserCredentials

"HashedUserCredentials" is for protected PDFs only and only if the PDF login form is set to use a hashed password, which by convention, is only used by legacy customers.

For recently protected PDFs (since ~2013) it uses UserCredentials, when a user enters a username and password. This means there is generally no need to handle this case.

SsoLiteToken

"SsoLiteToken" is used by a protected PDF and only when the user opens the file. It is known internally as "ping on start up" where the user would have previously been authenticated and now the cached user info is being used to re-authorize this user for this content.

When Type is “SsoLiteToken”, the Authenticate Endpoint needs to handle two situations **in the following order:**

Token has a string, Document has a value	Token contains the reader’s UserId, so check whether the UserId is valid and that the reader has access to the document specified.
Token has a string, Document is null	Token contains the reader’s UserId, so check whether the UserId is valid.

UniqueDocCopyIdToken

"UniqueDocCopyIdToken" is used by a version/unique protected PDF, also during the "ping on start up" event when they open the file.

When Type is “UniqueDocCopyIdToken”, the Authenticate Endpoint should use the same code as “UserCredentials”

Appendix A (Authenticate Endpoint Request Payload Samples)

Unlock UserCredentials PDF with Username Password

Type: UserCredentials

Key Fields: Username, Password, Document


```
{
  "Username": "user@domain.com",
  "Id": null,
  "Password": "********",
  "HashingKey": null,
  "HashingVersion": null,
  "CaseSensitivePassword": true,
  "Token": null,
  "Type": "UserCredentials",
  "Document": {
    "FolderPath": [
      "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
    ],
    "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
    "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
    "DocCode": "0000-2C30-1BF222-002675A3",
    "Metadata": {
      "ContentType": "std",
      "Title": "testdoc-multipage",
      "VersionName": null,
      "UserSpecificWatermarkTemplates": [
        {
          "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
          "Name": "default",
          "TextTemplate": "Licensed to _userName_"
        }
      ]
    },
    "ExternalKey": "166",
    "Status": {
      "IsActive": true,
      "IsMostRecentVersion": true,
      "IsMostRecentVersionActive": true
    },
    "Alias": "b92JcX"
  },
  "UserClient": {
    "AppName": "Reader",
    "AppVersion": "23.00120174",
    "Platform": "WIN",
```

```
    "OperatingSystem": "WIN",
    "DeviceName": null,
    "DeviceId": "{1684266512170-f5442874-c5f3-019f-0b05-fd2b63511bb4}",
    "HasOfflineAccess": false,
    "InjectVersion": "5_2_4",
    "IpAddress": "0.0.0.0",
    "Language": "ENU",
    "OutOfBrowser": true,
    "ServerUrl": "https://api.protectedpdf.com/"
  }
}
```


Unlock VersionUnique PDF

Type: UniqueDocCopyIdToken

Key Fields: Username, Document


```
{
  "Username": "user@domain.com",
  "Id": null,
  "Password": null,
  "HashingKey": null,
  "HashingVersion": null,
  "CaseSensitivePassword": false,
  "Token": null,
  "Type": "UniqueDocCopyIdToken",
  "Document": {
    "FolderPath": [
      "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
    ],
    "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
    "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
    "DocCode": "0000-2C30-1BF222-002675A3",
    "Metadata": {
      "ContentType": "std",
      "Title": "testdoc-multipage",
      "VersionName": null,
      "UserSpecificWatermarkTemplates": [
        {
          "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
          "Name": "default",
          "TextTemplate": "Licensed to _userName_"
        }
      ]
    },
    "ExternalKey": "166",
    "Status": {
      "IsActive": true,
```

```
        "IsMostRecentVersion": true,
        "IsMostRecentVersionActive": true
      },
      "Alias": "b92JcX"
    },
    "UserClient": {
      "AppName": "Reader",
      "AppVersion": "23.00120174",
      "Platform": "WIN",
      "OperatingSystem": "WIN",
      "DeviceName": null,
      "DeviceId": "{1684266512170-f5442874-c5f3-019f-0b05-fd2b63511bb4}",
      "HasOfflineAccess": false,
      "InjectVersion": "5_2_4",
      "IpAddress": "0.0.0.0",
      "Language": "ENU",
      "OutOfBrowser": true,
      "ServerUrl": "https://api.protectedpdf.com/"
    }
}
```


Unlock SSO PDF

Type: UniqueDocCopyIdToken

Key Fields: Username, Document


```
{
    "Username": "user@domain.com",
    "Id": null,
    "Password": null,
    "HashingKey": null,
    "HashingVersion": null,
    "CaseSensitivePassword": false,
    "Token": null,
    "Type": "UniqueDocCopyIdToken",
    "Document": {
      "FolderPath": [
        "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
        "ContentType": "std",
        "Title": "testdoc-multipage",
        "VersionName": null,
        "UserSpecificWatermarkTemplates": [
          {
```

```
            "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
            "Name": "default",
            "TextTemplate": "Licensed to _userName_"
          }
        ]
      },
      "ExternalKey": "166",
      "Status": {
        "IsActive": true,
        "IsMostRecentVersion": true,
        "IsMostRecentVersionActive": true
      },
      "Alias": "b92JcX"
    },
    "UserClient": {
      "AppName": "Reader",
      "AppVersion": "23.00120174",
      "Platform": "WIN",
      "OperatingSystem": "WIN",
      "DeviceName": null,
      "DeviceId": "{1684266512170-f5442874-c5f3-019f-0b05-fd2b63511bb4}",
      "HasOfflineAccess": false,
      "InjectVersion": "5_2_4",
      "IpAddress": "0.0.0.0",
      "Language": "ENU",
      "OutOfBrowser": true,
      "ServerUrl": "https://api.protectedpdf.com/"
    }
}
```


Unlock Regular PDF with SSO Lite

Type: SsoLiteToken
Key Fields: Token, Document


```
{
    "Username": null,
    "Id": null,
    "Password": null,
    "HashingKey": null,
    "HashingVersion": null,
    "CaseSensitivePassword": false,
    "Token": "user@domain.com",
    "Type": "SsoLiteToken",
    "Document": {
      "FolderPath": [
        "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
```

```
    ],
    "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
    "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
    "DocCode": "0000-2C30-1BF222-002675A3",
    "Metadata": {
      "ContentType": "std",
      "Title": "testdoc-multipage",
      "VersionName": null,
      "UserSpecificWatermarkTemplates": [
        {
          "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
          "Name": "default",
          "TextTemplate": "Licensed to _userName_"
        }
      ]
    },
    "ExternalKey": "166",
    "Status": {
      "IsActive": true,
      "IsMostRecentVersion": true,
      "IsMostRecentVersionActive": true
    },
    "Alias": "b92JcX"
  },
  "UserClient": {
    "AppName": "Reader",
    "AppVersion": "23.00120174",
    "Platform": "WIN",
    "OperatingSystem": "WIN",
    "DeviceName": null,
    "DeviceId": "{1684266512170-f5442874-c5f3-019f-0b05-fd2b63511bb4}",
    "HasOfflineAccess": false,
    "InjectVersion": "5_2_4",
    "IpAddress": "0.0.0.0",
    "Language": "ENU",
    "OutOfBrowser": true,
    "ServerUrl": "https://api.protectedpdf.com/"
  }
}
```


Unlock Web with Username/Password

Type: UserCredentials
Key Fields: Username, Password, Document


```
{
  "Username": "user@domain.com",
```

```
  "Id": null,
  "Password": "********",
  "HashingKey": null,
  "HashingVersion": null,
  "CaseSensitivePassword": true,
  "Token": null,
  "Type": "UserCredentials",
  "Document": {
    "FolderPath": [
      "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
    ],
    "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
    "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
    "DocCode": "0000-2C30-1BF222-002675A3",
    "Metadata": {
      "ContentType": "std",
      "Title": "testdoc-multipage",
      "VersionName": "Version: 1",
      "UserSpecificWatermarkTemplates": [
        {
          "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
          "Name": "default",
          "TextTemplate": "Licensed to _userName_"
        }
      ]
    },
    "ExternalKey": "166",
    "Status": {
      "IsActive": true,
      "IsMostRecentVersion": true,
      "IsMostRecentVersionActive": true
    },
    "Alias": "b92JcX"
  },
  "UserClient": {
    "AppName": "Chrome",
    "AppVersion": "113.0",
    "Platform": null,
    "OperatingSystem": "Windows 10",
    "DeviceName": "Other",
    "DeviceId": "WV-c604b57a-b6e3-4cc8-81f1-be5882b037ef",
    "HasOfflineAccess": false,
    "InjectVersion": null,
    "IpAddress": "0.0.0.0",
    "Language": "en",
    "OutOfBrowser": false,
    "ServerUrl": "https://view.protectedpdf.com/api/1.0/login/b92JcX"
```

```
    }
}
```


Unlock Web by Query Parm

Type: WebViewerSso

Key Fields: Token, Document


```
{
  "Username": null,
  "Id": null,
  "Password": null,
  "HashingKey": null,
  "HashingVersion": null,
  "CaseSensitivePassword": false,
  "Token": "ZG9hMGtvcFdkWVQraTVGdnFvdHH5dA9UZGNmZHhDdzlWalhMRlYxaWFDWT0=",
  "Type": "WebViewerSso",
  "Document": {
    "FolderPath": [
      "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
    ],
    "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
    "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
    "DocCode": "0000-2C30-1BF222-002675A3",
    "Metadata": {
      "ContentType": "std",
      "Title": "testdoc-multipage",
      "VersionName": "Version: 1",
      "UserSpecificWatermarkTemplates": [
        {
          "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
          "Name": "default",
          "TextTemplate": "Licensed to _userName_"
        }
      ]
    },
    "ExternalKey": "166",
    "Status": {
      "IsActive": true,
      "IsMostRecentVersion": true,
      "IsMostRecentVersionActive": true
    },
    "Alias": "b92JcX"
  },
  "UserClient": {
    "AppName": "Chrome",
```

```
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/b92JcX?token=
ZG9hMGtvcFdkWVQraTVGdnFvdHH5dA9UZGNmZHhDdzlWalhMRlYxaWFDWT0=&_sw=18826423d02"""
   }
}
```


Unlock Web by Query Parm (force authentication by token)

Type: WebViewerSso

Key Fields: Token, Document


```
{
   "Username": null,
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": "ZG9hMGtvcFdkWVQraTVGdnFvdJE5dm9yZGNmZHsDdzlW5lhMRlYxaWFDWT0=",
   "Type": "WebViewerSso",
   "Document": {
      "FolderPath": [
         "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
         "ContentType": "std",
         "Title": "testdoc-multipage",
         "VersionName": "Version: 1",
         "UserSpecificWatermarkTemplates": [
            {
               "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
               "Name": "default",
               "TextTemplate": "Licensed to _userName_"
            }
         ]
```

```
      },
      "ExternalKey": "166",
      "Status": {
         "IsActive": true,
         "IsMostRecentVersion": true,
         "IsMostRecentVersionActive": true
      },
      "Alias": "b92JcX"
   },
   "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"      ""ServerUrl"": ""https://view.protectedpdf.com/b92JcX?token=
ZG9hMGtvcFdkWVQraTVGdnFvdJE5dm9yZGNmZHsDdzlW5lhMRlYxaWFDWT0=&_sw=18826468e02"""
   }
}
```


Unlock Web by Cookie

Type: WebViewerSso
Key Fields: Token, Document


```
{
   "Username": null,
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": "ZG9hMGtvcFdkWVQraTVGdnFvdJE5dm9yZGNmZHsDdzlW5lhMRlYxaWFDWT0=",
   "Type": "WebViewerSso",
   "Document": {
      "FolderPath": [
         "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
```

```
      "Metadata": {
        "ContentType": "std",
        "Title": "testdoc-multipage",
        "VersionName": "Version: 1",
        "UserSpecificWatermarkTemplates": [
          {
            "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
            "Name": "default",
            "TextTemplate": "Licensed to _userName_"
          }
        ]
      },
      "ExternalKey": "166",
      "Status": {
        "IsActive": true,
        "IsMostRecentVersion": true,
        "IsMostRecentVersionActive": true
      },
      "Alias": "b92JcX"
    },
    "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/b92JcX?token=
ZG9hMGtvcFdkWVQraTVGdnFvdJE5dm9yZGNmZHsDdzlW5lhMRlYxaWFDWT0=&_sw=18826468e02"""
    }
}
```


Unlock Web by OAuth

Type: WebViewerSso
Key Fields: Username, Document


```
{
  "Username": "user@domain.com",
  "Id": null,
  "Password": null,
  "HashingKey": null,
```

```
  "HashingVersion": null,
  "CaseSensitivePassword": false,
  "Token": null,
  "Type": "WebViewerSso",
  "Document": {
    "FolderPath": [
      "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
    ],
    "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
    "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
    "DocCode": "0000-2C30-1BF222-002675A3",
    "Metadata": {
      "ContentType": "std",
      "Title": "testdoc-multipage",
      "VersionName": "Version: 1",
      "UserSpecificWatermarkTemplates": [
        {
          "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
          "Name": "default",
          "TextTemplate": "Licensed to _userName_"
        }
      ]
    },
    "ExternalKey": "166",
    "Status": {
      "IsActive": true,
      "IsMostRecentVersion": true,
      "IsMostRecentVersionActive": true
    },
    "Alias": "b92JcX"
  },
  "UserClient": {
    "AppName": "Chrome",
    "AppVersion": "113.0",
    "Platform": null,
    "OperatingSystem": "Windows 10",
    "DeviceName": "Other",
    "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
    "HasOfflineAccess": false,
    "InjectVersion": null,
    "IpAddress": "0.0.0.0",
    "Language": "en",
    "OutOfBrowser": false,
    "ServerUrl": "https://view.protectedpdf.com/OAuthSignIn?code="
  }
}
```

Unlock Web by Previous Session

Type: WebViewerSessionTokenVerification

Key Fields: Username, Document


```
{
  "Username": "user@domain.com",
  "Id": null,
  "Password": null,
  "HashingKey": null,
  "HashingVersion": null,
  "CaseSensitivePassword": false,
  "Token": null,
  "Type": "WebViewerSessionTokenVerification",
  "Document": {
    "FolderPath": [
      "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
    ],
    "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
    "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
    "DocCode": "0000-2C30-1BF222-002675A3",
    "Metadata": {
      "ContentType": "std",
      "Title": "testdoc-multipage",
      "VersionName": "Version: 1",
      "UserSpecificWatermarkTemplates": [
        {
          "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
          "Name": "default",
          "TextTemplate": "Licensed to _userName_"
        }
      ]
    },
    "ExternalKey": "166",
    "Status": {
      "IsActive": true,
      "IsMostRecentVersion": true,
      "IsMostRecentVersionActive": true
    },
    "Alias": "b92JcX"
  },
  "UserClient": {
    "AppName": "Chrome",
    "AppVersion": "113.0",
    "Platform": null,
    "OperatingSystem": "Windows 10",
    "DeviceName": "Other",
    "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
```

```
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"      ""ServerUrl"": ""https://view.protectedpdf.com/api/doc/b92JcX/
info?contentType=xod&count=0"""
   }
}
```


Web Viewer 5 minutes Ping

Type: WebViewerSessionTokenVerification

Key Fields: Username, Document


```
{
   "Username": "user@domain.com",
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": null,
   "Type": "WebViewerSessionTokenVerification",
   "Document": {
      "FolderPath": [
         "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
         "ContentType": "std",
         "Title": "testdoc-multipage",
         "VersionName": "Version: 1",
         "UserSpecificWatermarkTemplates": [
            {
               "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
               "Name": "default",
               "TextTemplate": "Licensed to _userName_"
            }
         ]
      },
      "ExternalKey": "166",
      "Status": {
         "IsActive": true,
         "IsMostRecentVersion": true,
```

```
          "IsMostRecentVersionActive": true
       },
       "Alias": "b92JcX"
    },
    "UserClient": {
       "AppName": "Chrome",
       "AppVersion": "113.0",
       "Platform": null,
       "OperatingSystem": "Windows 10",
       "DeviceName": "Other",
       "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
       "HasOfflineAccess": false,
       "InjectVersion": null,
       "IpAddress": "0.0.0.0",
       "Language": "en",
       "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/api/doc/b92JcX/
info?contentType=xod&count=25"""
    }
}
```


Web Viewer 5 minutes Ping (force authentication by token, query parameter)

Type: WebViewerSso

Key Fields: Token, Document


```
{
   "Username": null,
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": "ZG9hMGtvcFdkWVQraTVGdnFvdJE5dm9yZGNmZHsDdzlW5lhMRlYxaWFDWT0=",
   "Type": "WebViewerSso",
   "Document": {
      "FolderPath": [
         "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
         "ContentType": "std",
         "Title": "testdoc-multipage",
         "VersionName": "Version: 1",
         "UserSpecificWatermarkTemplates": [
```

```
            {
               "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
               "Name": "default",
               "TextTemplate": "Licensed to _userName_"
            }
         ]
      },
      "ExternalKey": "166",
      "Status": {
         "IsActive": true,
         "IsMostRecentVersion": true,
         "IsMostRecentVersionActive": true
      },
      "Alias": "b92JcX"
   },
   "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/b92JcX?token=
ZG9hMGtvcFdkWVQraTVGdnFvdJE5dm9yZGNmZHsDdzlW5lhMRlYxaWFDWT0=&_sw=18826468e02"""
   }
}
```


Download PDF for Print (Actual download button)

Type: PrintMeteringUsernameToken
Key Fields: Username, Document


```
{
   "Username": "user@domain.com",
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": null,
   "Type": "PrintMeteringUsernameToken",
   "Document": {
```

```
      "FolderPath": [
        "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
        "ContentType": "doc",
        "Title": "testdoc-multipage",
        "VersionName": "Version: 1",
        "UserSpecificWatermarkTemplates": null
      },
      "ExternalKey": "166",
      "Status": null,
      "Alias": "b92JcX"
    },
    "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
      "ServerUrl": null
    }
}
```


Web Print from Web Viewer

Type: WebViewerSessionTokenVerification

Key Fields: Username, Document


```
{
    "Username": "user@domain.com",
    "Id": null,
    "Password": null,
    "HashingKey": null,
    "HashingVersion": null,
    "CaseSensitivePassword": false,
    "Token": null,
    "Type": "WebViewerSessionTokenVerification",
    "Document": {
      "FolderPath": [
```

```
        "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
        "ContentType": "std",
        "Title": "testdoc-multipage",
        "VersionName": "Version: 1",
        "UserSpecificWatermarkTemplates": [
          {
            "Id": "2da8b09a-a7ad-4549-8948-bbc58bc654f3",
            "Name": "default",
            "TextTemplate": "Licensed to _userName_"
          }
        ]
      },
      "ExternalKey": "166",
      "Status": {
        "IsActive": true,
        "IsMostRecentVersion": true,
        "IsMostRecentVersionActive": true
      },
      "Alias": "b92JcX"
    },
    "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": null,
      "OutOfBrowser": false,
      "ServerUrl": "https://view.protectedpdf.com/api/doc/b92JcX/WebPrint"
    }
}
```


Unlock Portal with Username/Password

Type: UserCredentials
Key Fields: Username, Password


```
{
  "Username": "user@domain.com",
```

```
  "Id": null,
  "Password": "********",
  "HashingKey": null,
  "HashingVersion": null,
  "CaseSensitivePassword": false,
  "Token": null,
  "Type": "UserCredentials",
  "Document": null,
  "UserClient": {
    "AppName": "Chrome",
    "AppVersion": "113.0",
    "Platform": null,
    "OperatingSystem": "Windows 10",
    "DeviceName": "Other",
    "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
    "HasOfflineAccess": false,
    "InjectVersion": null,
    "IpAddress": "0.0.0.0",
    "Language": "en",
    "OutOfBrowser": false,
    "ServerUrl": "https://view.protectedpdf.com/api/portal/vit-daniel/login/"
  }
}
```


Unlock Portal SSO by Query Param

Type: WebViewerSso

Key Fields: Token


```
{
  "Username": null,
  "Id": null,
  "Password": null,
  "HashingKey": null,
  "HashingVersion": null,
  "CaseSensitivePassword": false,
  "Token": "ZG9hMGtvcFdkWrQraTVGdnFvdHE5dm9yzGhmZHhDdzlWalhMRlYxaWFDWT0=",
  "Type": "WebViewerSso",
  "Document": null,
  "UserClient": {
    "AppName": "Chrome",
    "AppVersion": "113.0",
    "Platform": null,
    "OperatingSystem": "Windows 10",
    "DeviceName": "Other",
    "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
    "HasOfflineAccess": false,
```

```
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/portal/vit-daniel?
token=ZG9hMGtvcFdkWrQraTVGdnFvdHE5dm9yzGhmZHhDdzlWalhMRlYxaWFDWT0=&_sw=1882688fb47"""
   }
}
```


Unlock Portal SSO by Query Parm (force authentication)

Type: WebViewerSso
Key Fields: Token


```
{
   "Username": null,
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": "ZG9hMGtvcFdkWrQraTVGdnFvdHE5dm9yzGhmZHhDdzlWalhMRlYxaWFDWT0=",
   "Type": "WebViewerSso",
   "Document": null,
   "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/portal/vit-daniel?
token=ZG9hMGtvcFdkWrQraTVGdnFvdHE5dm9yzGhmZHhDdzlWalhMRlYxaWFDWT0=&_sw=1882688fb47"""
   }
}
```


Unlock Portal SSO by Cookie

Type: WebViewerSso
Key Fields: Token

```
{
   "Username": null,
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": "ZG9hMGtvcFdkWrQraTVGdnFvdHE5dm9yzGhmZHhDdzlWalhMRlYxaWFDWT0=",
   "Type": "WebViewerSso",
   "Document": null,
   "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/portal/vit-daniel?
token=ZG9hMGtvcFdkWrQraTVGdnFvdHE5dm9yzGhmZHhDdzlWalhMRlYxaWFDWT0=&_sw=1882688fb47"""
   }
}
```


Unlock Portal SSO by OAuth

Type: WebViewerSso
Key Fields: Username


```
{
   "Username": "user@domain.com",
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": null,
   "Type": "WebViewerSso",
   "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
```

```
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
      "ServerUrl": "https://view.protectedpdf.com/OAuthSignIn?code="
   }
}
```


Portal 5 minutes Ping

Type: WebViewerSessionTokenVerification

Key Fields: Username


```
{
   "Username": "user@domain.com",
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": null,
   "Type": "WebViewerSessionTokenVerification",
   "Document": null,
   "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": null,
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/api/portal/vit-daniel/
PortalChange/?_=1684271798711"""
   }
}
```


Unlock Portal by previous session

Type: WebViewerSessionTokenVerification

Key Fields: Username


```
{
   "Username": "user@domain.com",
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": null,
   "Type": "WebViewerSessionTokenVerification",
   "Document": null,
   "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": null,
      "OutOfBrowser": false,
"       ""ServerUrl"": ""https://view.protectedpdf.com/api/portal/vit-daniel/
PortalChange/?_=1684271798711"""
   }
}
```


Download Regular PDF from Portal [Deprecated in v3.0]

Type: DownloadProtectedUsernameToken (External Service v3.5 only)
Key Fields: Username, Document


```
{
   "Username": "user@domain.com",
   "Id": null,
   "Password": null,
   "HashingKey": null,
   "HashingVersion": null,
   "CaseSensitivePassword": false,
   "Token": null,
   "Type": "DownloadProtectedUsernameToken",
   "Document": {
      "FolderPath": [
         "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
```

```
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
        "ContentType": "doc",
        "Title": "testdoc-multipage",
        "VersionName": "Version: 1",
        "UserSpecificWatermarkTemplates": null
      },
      "ExternalKey": "166",
      "Status": null,
      "Alias": "b92JcX"
    },
    "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "113.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-d08a214b-d3e8-4acf-ac77-0426b9180301",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
      "ServerUrl": null
    }
}
```


Download SSO PDF from Portal [Deprecated in v3.0]

Type: DownloadUniqueUsernameToken (External Service v3.5 only)

Key Fields: Username, Document


```
{
    "Username": "user@domain.com",
    "Password": null,
    "HashingKey": null,
    "HashingVersion": null,
    "CaseSensitivePassword": false,
    "Token": null,
    "Type": "DownloadUniqueUsernameToken",
    "Document": {
      "FolderPath": [
        "03c6be79-3495-4deb-bcb3-29a991468976"
      ],
      "DocumentId": "d0511f66-668a-4b27-9f34-0996f79173c1",
      "VersionId": "5b09a081-d471-488c-8ca3-5a5bc8e0d27b",
      "DocCode": "0000-2BB8-1A6CB8-0024AF3E",
      "Metadata": {
```

```
        "ContentType": "doc",
        "Title": "Handout 1",
        "Alias": null,
        "VersionName": "Version: 1",
        "UserSpecificWatermarkTemplates": null
      },
      "ExternalKey": "ValidExternalKey",
      "Status": null
    },
    "UserClient": {
      "AppName": "Chrome",
      "AppVersion": "112.0",
      "Platform": null,
      "OperatingSystem": "Windows 10",
      "DeviceName": "Other",
      "DeviceId": "WV-93c2a7d2-d276-4262-9712-955dd10dafc2",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": "0.0.0.0",
      "Language": "en",
      "OutOfBrowser": false,
      "ServerUrl": null
    }
}
```


PhoneUnlockToken

Type: PhoneUnlockToken

Key Fields: Username, Document


```
{
    "Username": "user@domain.com",
    "Id": null,
    "Password": null,
    "HashingKey": null,
    "HashingVersion": null,
    "CaseSensitivePassword": false,
    "Token": null,
    "Type": "PhoneUnlockToken",
    "Document": {
      "FolderPath": [
        "57b1ab7a-0c9c-4847-8564-868fdbccce5e"
      ],
      "DocumentId": "dba6d867-dcae-4064-81d7-d37c35f16e7c",
      "VersionId": "14b12230-1832-4197-ab1c-7f3ab46b94ea",
      "DocCode": "0000-2C30-1BF222-002675A3",
      "Metadata": {
        "ContentType": null,
```

```
        "Title": "testdoc-multipage",
        "VersionName": null,
        "UserSpecificWatermarkTemplates": null
      },
      "ExternalKey": "166",
      "Status": {
        "IsActive": true,
        "IsMostRecentVersion": false,
        "IsMostRecentVersionActive": false
      },
      "Alias": null
    },
    "UserClient": {
      "AppName": null,
      "AppVersion": null,
      "Platform": null,
      "OperatingSystem": null,
      "DeviceName": null,
      "DeviceId": "Remote-unlocktest",
      "HasOfflineAccess": false,
      "InjectVersion": null,
      "IpAddress": null,
      "Language": null,
      "OutOfBrowser": false,
      "ServerUrl": null
    }
}
```

Appendix B (Permissions Endpoint Response Samples)

Retrieve Single Content with DocExternalKeys[]

The portal will display a content with matching external key

```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": [
    "100"
  ],
  "FolderExternalKeys": []
}
```

Retrieve Multiple Content with DocExternalKeys[]

The portal will display contents with matching external keys

```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": [
    "100", “101”, “102”
  ],
  "FolderExternalKeys": []
}
```

Retrieve Multiple Content with DocIncludeExternalKeys []

The portal will display contents with external keys that contains the specified keywords

```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": [],
  "FolderExternalKeys": [],
  "DocIncludeExternalKeys": [";Bronze;"]
}
```

This example will return contents with ExternalKey "Silver;Bronze;DOC-2" and "Gold;Silver;Bronze;DOC-2" as both of them contains the keyword ";Bronze;"

Retrieve Single Content with DocIds[]

The portal will display a content with matching doc id

```
{
  "DocIds": [“d0b1c40a-778e-3120-9b84-1f7673627ec3”],
  "FolderIds": [],
  "DocExternalKeys": [],
  "FolderExternalKeys": []
}
```

Retrieve Multiple Content with DocIds []

The portal will display contents with matching doc ids

```
{
  "DocIds": [“d0b1c40a-778e-3120-9b84-1f7673627ec3”, “d0b2c30a-778e-6120-9b856-2f7674627ec3”],
  "FolderIds": [],
  "DocExternalKeys": [],
  "FolderExternalKeys": []
}
```

Retrieve All Contents inside a folder with FolderExternalKeys []

The portal will display all contents inside a folder with matching folder external key

```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": [],
  "FolderExternalKeys": [
    “1000”
  ]
}
```

Retrieve All Contents inside folders with FolderExternalKeys []

The portal will display all contents inside folders with matching folder external keys

```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": [],
  "FolderExternalKeys": [ “1000”, “1001”, “1002” ]
}
```

Retrieve All Contents inside folders with FolderIncludeExternalKeys []

The portal will display all contents inside folders with external keys that contains the specified keywords

```
{
  "DocIds": [],
  "FolderIds": [],
  "DocExternalKeys": [],
  "FolderExternalKeys": [],
  "FolderIncludeExternalKeys": ["Member"]
}
```

This example will return all contents inside folders with ExternalKey "Silver Member" and "Gold Member" as both of them contains the keyword "Member"

Retrieve All Contents inside a folder with FolderIds []

The portal will display all contents inside a folder with matching folder external key

{
"DocIds": [],
"FolderIds": [“4ba3225e-edc7-488c-a278-b536ce1b3323”],
"DocExternalKeys": [],
"FolderExternalKeys": []
}

Retrieve All Contents inside folders with FolderIds []

The portal will display all contents inside folders with matching folder external keys

{
"DocIds": [],
"FolderIds": [“4ba3225e-edc7-488c-a278-b536ce1b3323”, “1cj3225f-hdc7-488c-a278-b536ce1c1143”],
"DocExternalKeys": [],
"FolderExternalKeys": []
}

*** All four fields can be used at the same time, if necessary, and it will return all documents with matching external keys & ids. Note that providing FolderExternalKeys and DocExternalKeys does not filter out by the DocExternalKeys. Instead, it will return all contents inside with a specified folder and content matching specified DocExternalKeys.*

*** We do not break down external keys by delimiters. The values you provide with DocIncludeExternalKeys and FolderIncludeExternalKeys are used in SQL LIKE '%{value}%' queries for a "contains" match.*

For example:
Input: "DocIncludeExternalKeys": ["10"]
Result: Matches keys like 10, 100, 1000, a10b, 400100, abc;123;50101;blabla
This means any key containing "10" will be included in the results, regardless of formatting.

It is recommended to use External Keys rather than Ids as it requires more challenges to configurate the external service with Ids.

Appendix C: Differences between API 3.0 and 3.5

- Changes to the AuthenticationRequest
  - Additional Field named Id that will contain the Id of the User
- Changes to behavior for the following AuthenticationRequest Type
  - SSOLiteToken: In 3.0, the Id of the User is sent through the Token field. In 3.5, the Id of the User is sent through the Id field.
  - Additional request Types to support downloading a Protected PDF:
    - DownloadUniqueUsernameToken: Sent when a user requests a SSO Protected PDF from the Web Viewer. When the user opens this type of Protected PDF, the transaction type for the unlock request will be UniqueDocCopyIdToken.
    - DownloadProtectedUsernameToken: Sent when a user requests a Regular PDF be download from the Web Viewer but a Protected PDF that requires Username/Password. When the user opens this type of Protected PDF, the transaction type for the unlock request will be UserCredentials.

All available data will be sent through

In version 3.5, all available data will be sent through whereas in version 3.0, only selected data will be sent through.

- For example, under a situation where a user is authenticated through Web Viewer SSO by token/cookie, the following fields will be sent through:
  - Token
  - Document
- After the user is authenticated, the following fields will be sent through:
  - Id
  - Username
  - Token
  - Document
- Given that Token may expire due to it being a one-time use or there might be a time-based expiry set on it, the order of processing matters

Username/Password

- Initially, they'll send username/password.
- Then they won't send password.

Processing Heuristic if you will be implementing the processing without using the Vitrium sample code

Processing Heuristic	Note
1. Check if Password exists	This is only required if you wish to allow users to enter their username/password in the web viewer and PDF
2. Check if Username exists	This is required for all scenarios
3. Check if Token exists	This is only required for SSO sign in using Token/Cookie in the Web Viewer

4. Check if Id exists	This is only required if you wish to allow users to enter their username/password in the PDF

Inside each step above, you should handle both Document containing data or null.