Vitrium API Guide for User Authentication, Authorization & SSO

For Vitrium Security v7.x.x/v8.0.x and JSON Server v2 Specification

Updated: October 3, 2019
Document Revision: 1.04

1.604.677.1500 (direct)
1.866.403.1500 (toll-free)

www.vitrium.com

Table of Contents

Document Revisions

Revision Number	Reason	Date
1.01	Created revision table	October 15, 2018
	Added “message” to authorize and authentication endpoints	
1.02	Fixed typos	June 18, 2019
1.03	Added External Auth and OAuth2 scenario	June 19, 2019
1.04	Get Reader By token does not need to be a GUID	October 3, 2019

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	RTF and TXT	JPG
Word	OpenOffice word processing	PNG
Excel	OpenOffice spreadsheets	BMP
PowerPoint	OpenOffice presentations	TIF

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	128-bit AES encryption	Viewed with Adobe Reader or Acrobat (PC or Mac desktop versions only)
Secured Web Link	256-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

Implementing Vitrium's JSON APIs

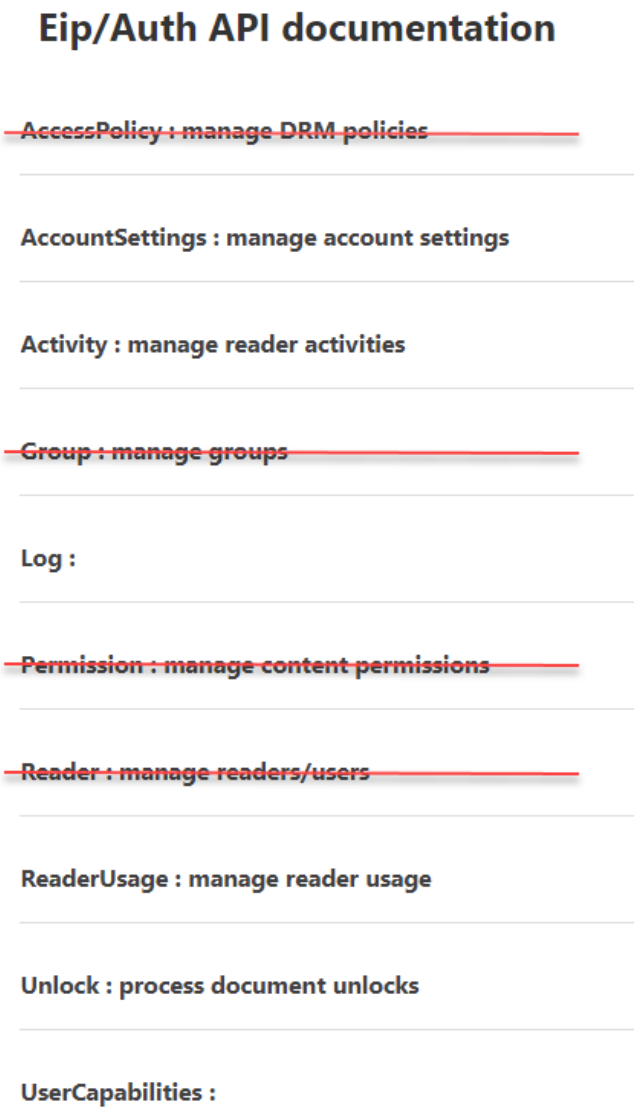
This next section walks you through all the various JSON-based APIs for the Vitrium software to talk to your systems for authentication, authorization and troubleshooting.

IMPORTANT!

It's important to understand that when implementing these JSON-based APIs that there are no "user" records (aka readers) stored inside the Vitrium system. "Users" seen in the Vitrium Admin User Interface (UI) are loaded in real-time using the Readers and/or Reader JASON API end points on your host. As such, do not utilize the Vitrium APIs in the sections identified to the right that are redlined as they will produce erroneous results.

Content permissions must be managed via your JSON API responses.

There is no concept of a Group at that point and activity reporting "by group" would need to occur on your side by utilizing "**Activity: manage reader activities**" API calls and linking those responses to "group" codes stored in your external EIP via the users id.



For User Authentication

This method authenticates the user with a username and password. It is primarily used for the manual unlock process of Vitrium-secured documents.

Request:

```
POST https://server/2.0/authenticate

JSON body:
{
       "UserName": "name@domain.com",
       "Password": "Test123"
}
```

Successful Response:

```
HTTP error 200 – OK with JSON in body
JSON example 1 with successful authentication:
{
       "Succeed": true
}
```

Failed Response:

```
HTTP error 200 – OK with JSON in body
JSON example 2 with failed authentication:
{
       "Succeed": false,
       "Message": "Optional. WARNING: If provided, this will force Succeed to be false"
}
```

For Authorizing Permission

This method verifies whether a user has permission to a particular document or image. This is used during manual and SSO lite unlocks.

Request:

```
POST https://server/2.0/authorize

JSON body:
{
	"ReaderId": "8d241bea-e714-4182-8257-01abfebff133",
	"DocCode": "0000-1111-22222-33333333",
	"DocId": "c91a3156-e1a9-4735-b18b-ce6733a6754c",
	"DocExternalKey": "DOC123",
	"ClientDeviceId": "{1487461123171-c97f1220-4875-938b-a5ca-db66796f3c2b}",
}
```

Attributes	Description
ReaderId	The user's ID
DocCode [1]	An internal ISBN-like document ID (stored with a Vitrium document or image)
DocId [2]	An internal GUID-based document ID (stored within your system)
DocExternalKey [3]	The customer's external document identifier
ClientDeviceId	A unique but random Adobe Reader/Acrobat specific device ID (not GUID but a string)

[1] The file will first need to be uploaded into Vitrium to retrieve the unique DocCode and it follows the form of 0000-0000-00000-00000000.
[2] Again, the file will first need to be uploaded into Vitrium but you can store a GUID-based ID within your system.
[3] If using the DocId, you will need to enter this as a value for DocExternalKey.

Successful Response:

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

Failed Response:

```
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",
	"AccessPolicy": null
}
```

To Retrieve a List of Users

This API call is primarily used when an administrator wants to view all users in the Vitrium admin interface. This is also a good way to view user activity reports, to clear past use, or to override a DRM policy.

Requests:

```
GET https://server/2.0/readers

GET https://server/2.0/readers?page={"index":1,"size":20}&sort={"Username":1}

GET
https://server/2.0/readers?page={"index":1,"size":20}&filter={"contains":"searchterm"}&sort={"username"
:-1}
```

Attributes	Description	
Page	index	Page number (first page is one not zero).
	size	Number of items per page (default is 20)
Filter	contains	Search term to run your searches
Sort	username	Order by the username in the following order: For Ascending order, use 1 For Descending order, use -1

Successful Response:

```
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": {
			"ComputersMax": 2,
			"IpAddressesMax": null,
			"IpRangeAllowed": null,
			"OfflineDurationinDays": 7,
			"Expiry": "1900-01-01",
			"DocumentLimit": null,
			"RelativeExpiryInDays": null,
			"OpenLimit": null
		}
	},
	…
	],
	"TotalRecords": 25
}
```

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

Attributes	Description	Definition of Values
Id	Unique ID to identify a user	
Username	Unique username for a user to login to a document	
IsActive	Defines whether a user is active or not	True: active False: inactive
ComputersMax [4]	Total number of applications (Adobe PDF viewers and web browsers) that the user can unlock a document with *****IMPORTANT: review the footnote below*****	null: unlimited -1: value not defined
PdfLimit [4]	Total number of Adobe PDF viewers that a user can unlock the protected PDF document with	null: unlimited -1: value not defined
BrowserLimit [4]	Total number of unique web browsers that a user can unlock the secured web document with	null: unlimited -1: value not defined
OfflineDurationinDays	Number of days the user can view the document in offline mode	null: unlimited -1: value not defined
Expiry	Last date in which the user can access a secured document	null: unlimited 1900-01-01: value not defined
IpAddressesMax	Number of IP addresses that could be used to unlock a document	null: unlimited -1: value not defined
IgnoredIpAddresses	Range of IP addresses that could be used to unlock a document. The IP address defined in here will not count against the IpAddressesMax value.	null: not defined
DocumentLimit	Number of documents that the user can unlock	null: unlimited -1: value not defined
RelativeExpiryInDays	Number of days the user can unlock a document after the first unlock session	null: unlimited -1: value not defined
OpenLimit	Number of times the user can unlock a document after the first unlock session	null: unlimited -1: value not defined
WebPrintLimit	Number of times the user can print a copy of the web document in a browser	null: unlimited -1: value not defined
PrintLimit	Number of times the user can download a printable copy of a document	null: unlimited -1: value not defined

[4]. ComputersMax and PdfLimit / BrowserLimit are mutually exclusive. When setting ComputerMax, you have to set both PdfLimit and BrowserLimit to "-1" otherwise you will have confusing results. Likewise, if setting PdfLimit or BrowserLimit, then set ComputerMax to "-1".

To Retrieve a User by Username

This API call is to retrieve a user by their username or email address. It can be used for both the manual login and for the True SSO login (with the protected PDF only). As with the previous section, this is also a good way to view an activity report related to a particular user to troubleshoot any issues they may be encountering, to clear their past use, or to override a DRM policy.

Request:

```
GET https://server/2.0/reader?username=name@domain.com
```

Successful Response:

```
HTTP error 200 – OK with JSON in body
JSON example 1 with no policy override:
{
       "Id": "8d241bea-e714-4182-8257-01abfebff133",
       "Username": "name@domain.com",
       "IsActive": true,
       "AccessPolicy": null
}

JSON example 2 with specific policy override:
{
       "Id": "8d241bea-e714-4182-8257-01abfebff133",
       "Username": "name@domain.com",
       "IsActive": true,
       "AccessPolicy": {
              "ComputersMax": 2,
              "IpAddressesMax": null,
              "IpRangeAllowed": null,
              "OfflineDurationinDays": 7,
              "Expiry": "1900-01-01",
              "DocumentLimit": null,
              "RelativeExpiryInDays": null,
              "OpenLimit": null
       }
}
```

Failed Response:

```
HTTP error 404 – Not Found
```

To Retrieve a User by User ID

This API call is to retrieve a user by their user ID (for Lite SSO). This is also called when you want to look up a user in the Vitrium software to clear past use, override a policy, or view an activity report for troubleshooting purposes.

Request:

```
GET https://server/2.0/reader/12345
```

Successful Response:

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

JSON example 1 with no policy override:
{
       "Id": "8d241bea-e714-4182-8257-01abfebff133",
       "Username": "name@domain.com",
       "IsActive": true,
       "AccessPolicy": null
}

JSON example 2 with specific policy override:
{
       "Id": "8d241bea-e714-4182-8257-01abfebff133",
       "Username": "name@domain.com",
       "IsActive": true,
       "AccessPolicy": {
              "ComputersMax": 2,
              "IpAddressesMax": null,
              "IpRangeAllowed": null,
              "OfflineDurationinDays": 7,
              "Expiry": "1900-01-01",
              "DocumentLimit": null,
              "RelativeExpiryInDays": null,
              "OpenLimit": null
       }
}
```

Failed Response:

```
HTTP error 404 – Not Found
```

To Retrieve a User by SSO Token

This API call is to retrieve a user by their SSO token. A token is a string -- usually encoded -- and can contain any information or codes that you care to pass to the WebViewer session. Vitrium will pass into this JSON Server endpoint any token which is supplied as a query string parameter to a Web Viewer URL (eg. https://view.vitrium.com/AbCdEf?**token=*123***). This gives you the ability to generate the token in your site, or user portal, and include whatever information is useful to retrieve a user in your system.

Request:

```
GET https://server/2.0/reader?token=52f17b69-3a5b-46c4-a433-346c94f92e95
```

Successful Response:

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

JSON example 1 with no policy override:
{
       "Id": "8d241bea-e714-4182-8257-01abfebff133",
       "Username": "name@domain.com",
       "IsActive": true,
       "AccessPolicy": null
}

JSON example 2 with specific policy override:
{
       "Id": "8d241bea-e714-4182-8257-01abfebff133",
       "Username": "name@domain.com",
       "IsActive": true,
       "AccessPolicy": {
              "ComputersMax": 2,
              "IpAddressesMax": null,
              "IpRangeAllowed": null,
              "OfflineDurationinDays": 7,
              "Expiry": "1900-01-01",
              "DocumentLimit": null,
              "RelativeExpiryInDays": null,
              "OpenLimit": null
       }
}
```

Failed Response:

```
HTTP error 404 – Not Found
```

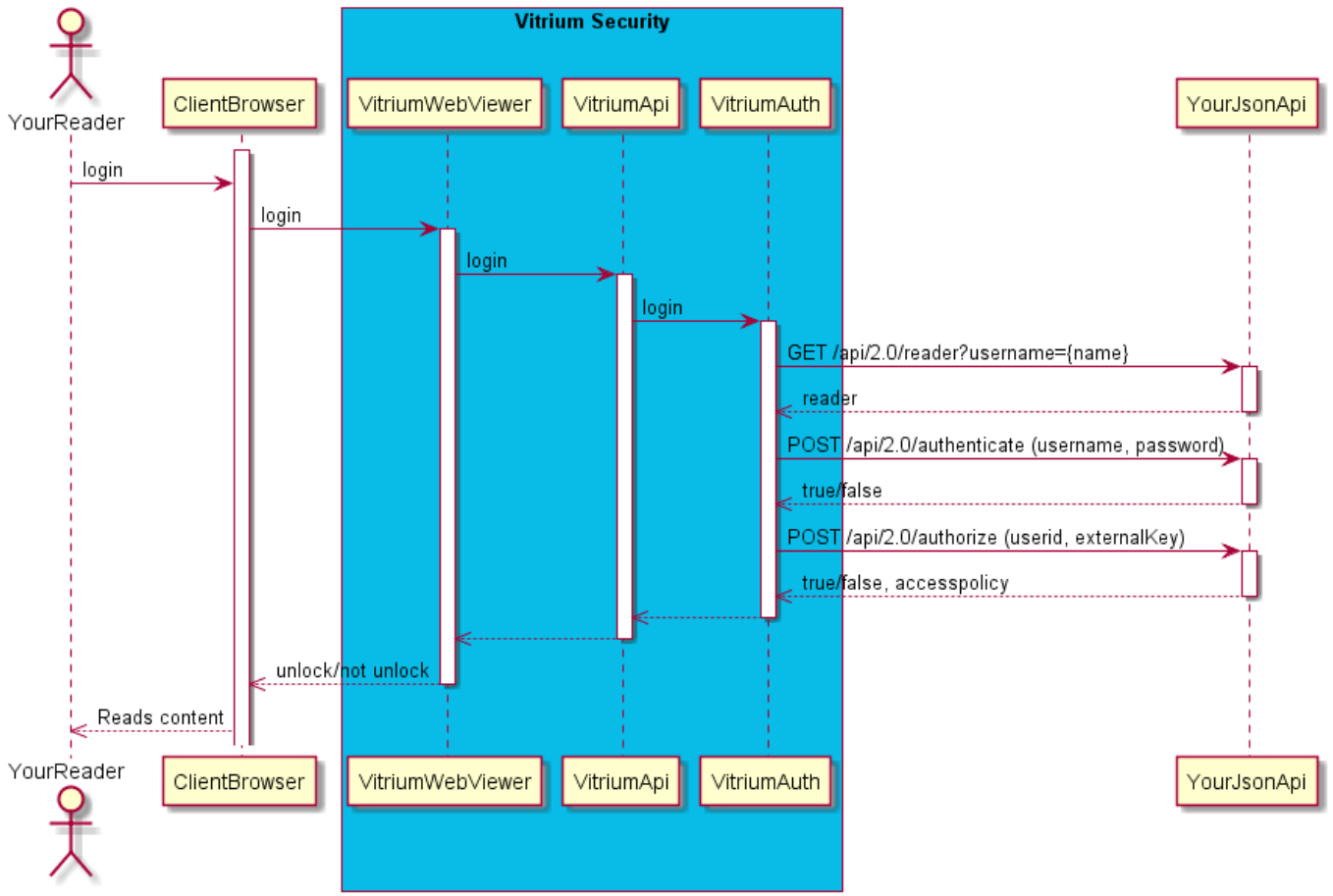
Sequence Diagrams & API Calls

For Manual Unlock

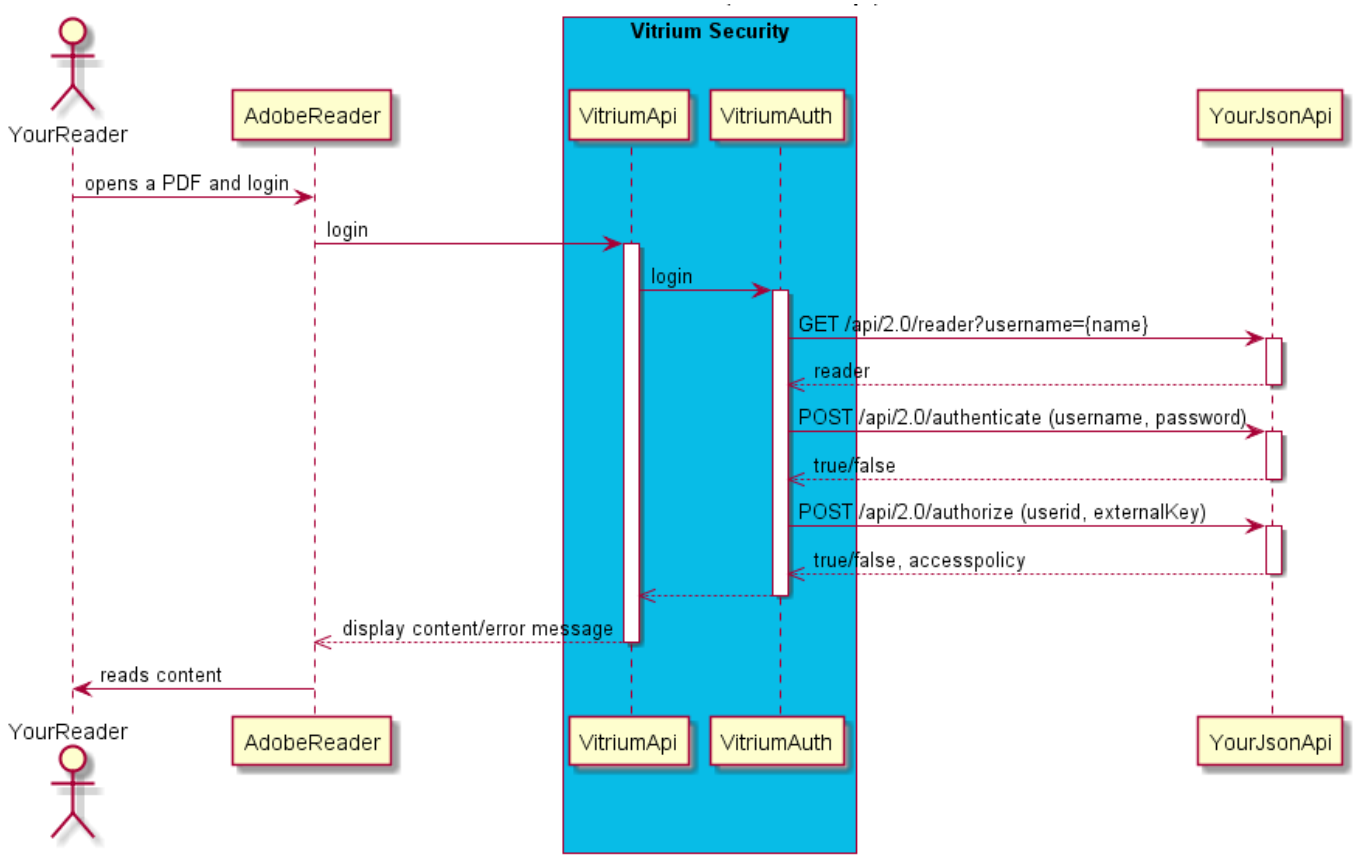
When a user (known as YourReader in this diagram) opens a protected PDF document or clicks on a secured web link, the user is prompted to enter their credentials. After that, the username and hashed password is transmitted to the Auth component. The Auth component will then perform the following outbound calls before deciding on whether to unlock the document.

Step	API Call	Purpose
1	GET /2.0/reader?username={name}	Get the Reader object for subsequent processing
2	POST /2.0/authenticate	Validate the userid (from step 1) and password
3	POST /2.0/authorize	Check if the userid (from step 1) has access to the document

Secured Web Link Process:



Protected PDF Process:

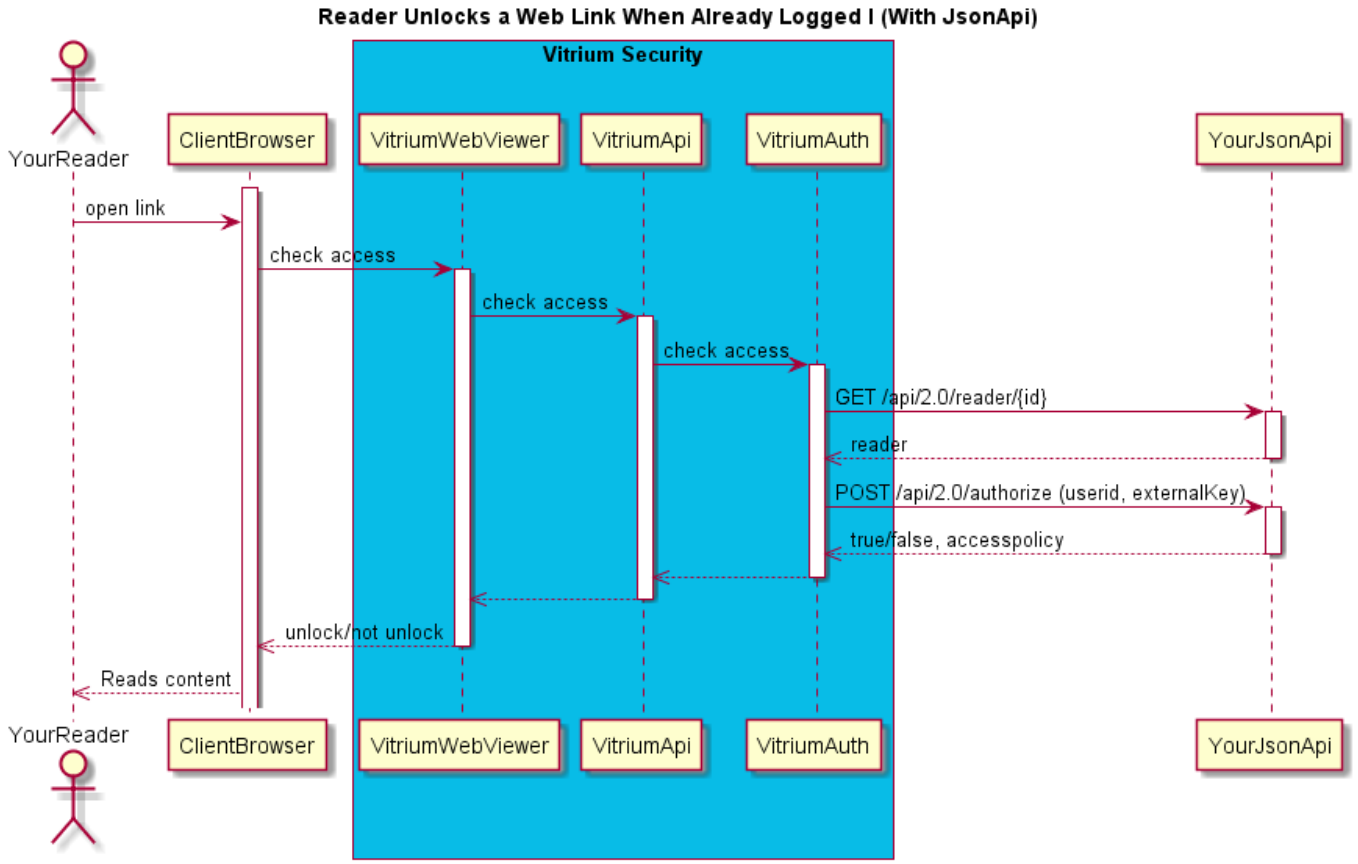


For Lite Single Sign-on (SSO) Unlock

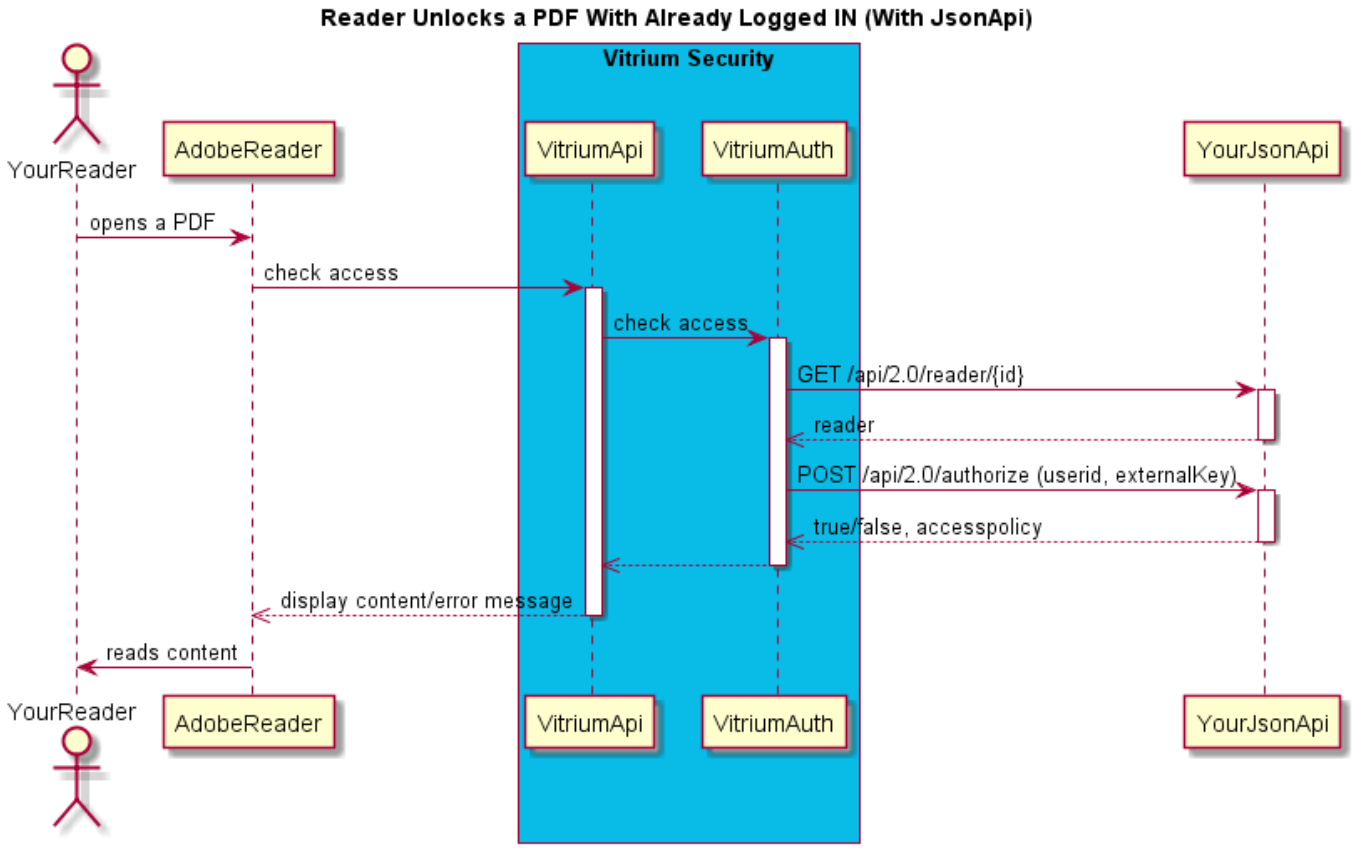
When a user clicks on a secured web link or opens a protected PDF file when SSO Lite is enabled, the user's device ID is sent to the Auth component. The Auth then looks up the user's ID based on this device ID. The Auth component will then perform the following outbound calls before deciding on whether to unlock the document.

Step	API Call	Purpose
1	GET /2.0/reader/{id}	Get the Reader object for subsequent processing
2	POST /2.0/authorize	Check if the userid (from step 1) has access to the document

Secured Web Link Process:



Protected PDF Process:



For True Single Sign-on (SSO) Unlock

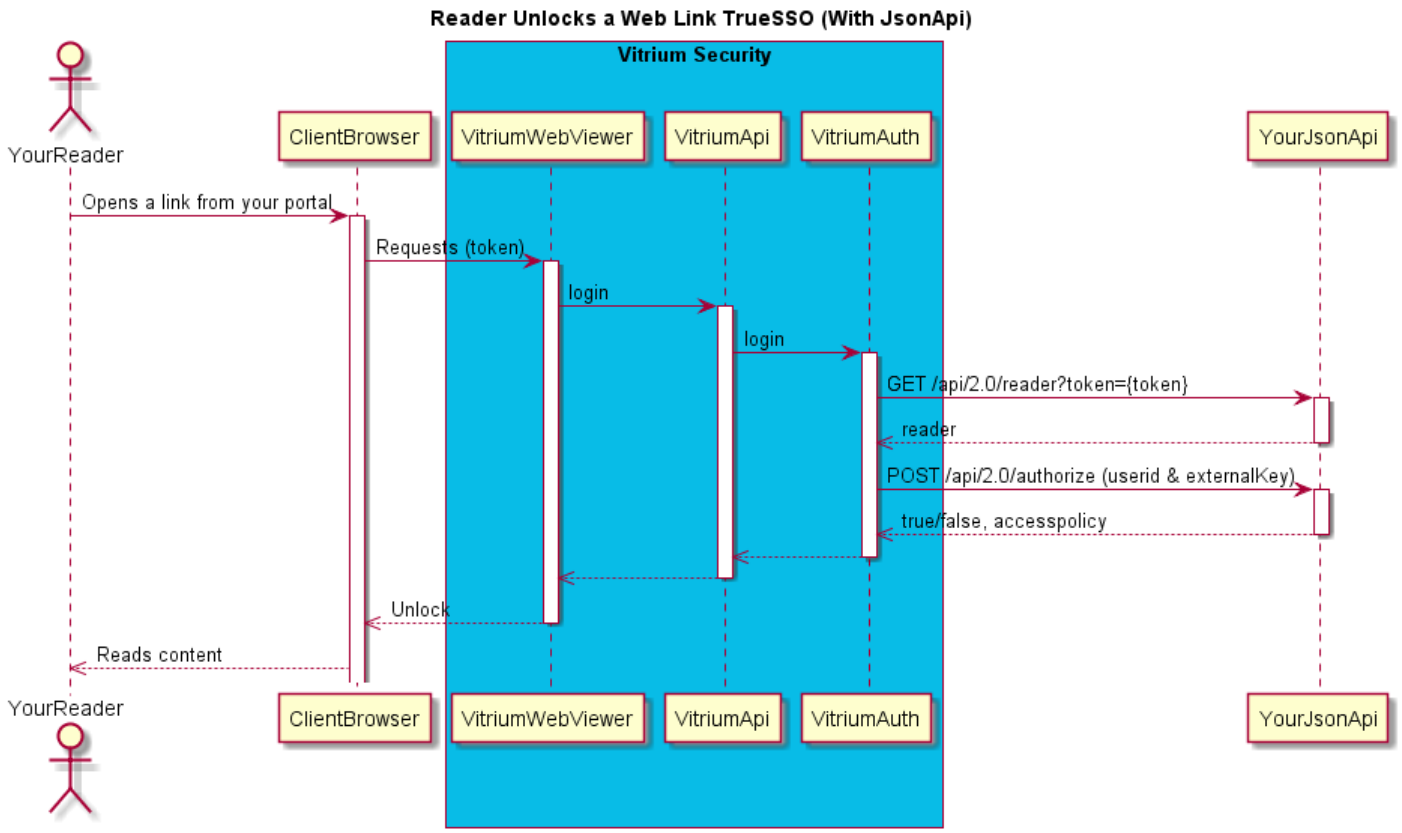
When a user opens a secured web link or a protected PDF document that has True SSO enabled, the user's session/token ID is sent to the Auth component. The Auth component will perform the following outbound calls before deciding on whether to unlock the document. The customer will need to keep track of sessions/tokens and the associated end users.

For the secured web links, either a predefined domain wide cookie or a unique URL parameter will need to be used.

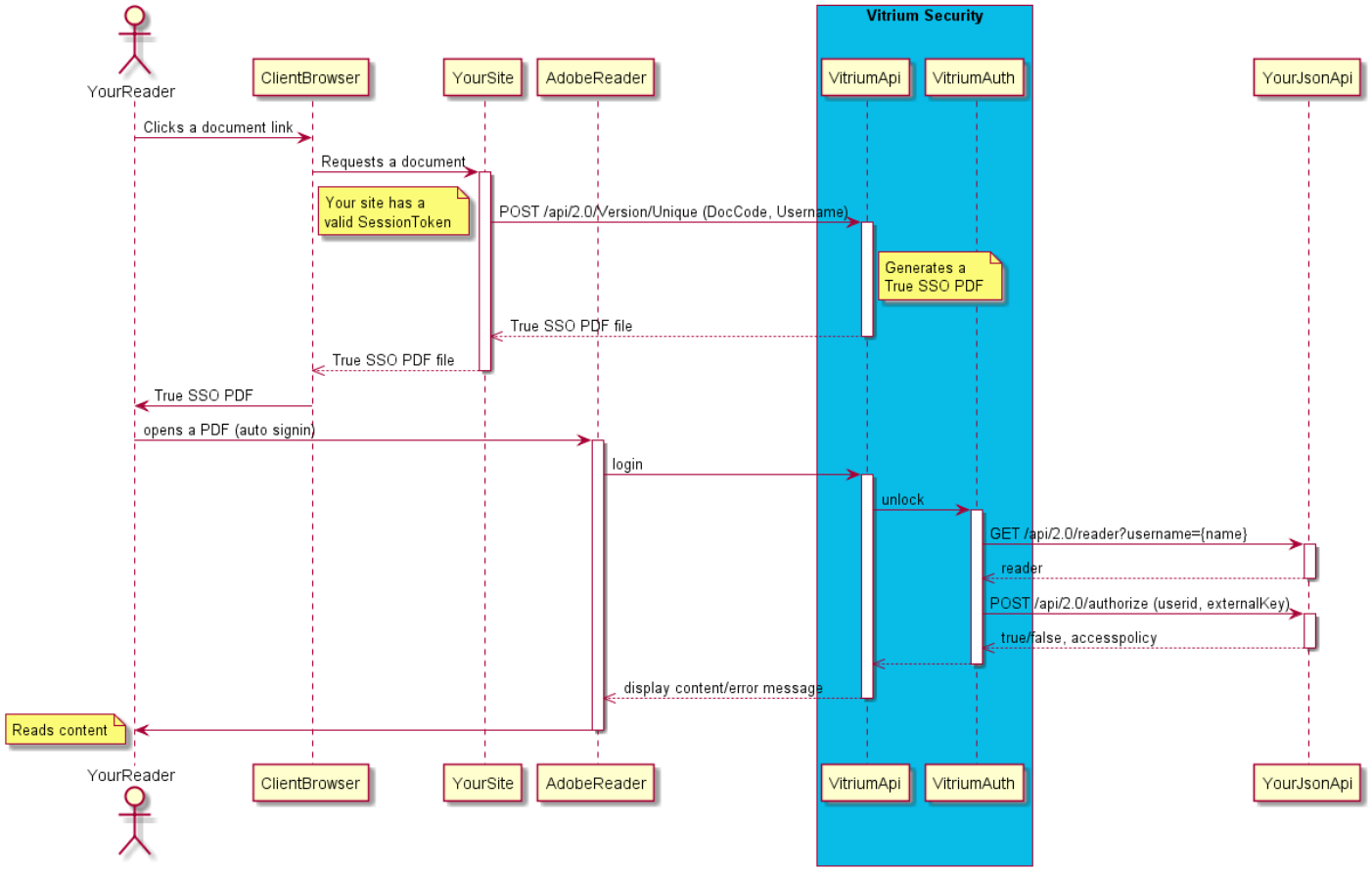
For the protected PDF files, a unique variant of the PDF will need to be created on demand and a session ID/token supplied at that time.

Step	API Call	Purpose
1	GET /2.0/reader?token={token}	Get the Reader object based on the session token
2	POST /2.0/authorize	Check if the userid (from step 1) has access to the document

Secured Web Link Process (External Auth/EIP):

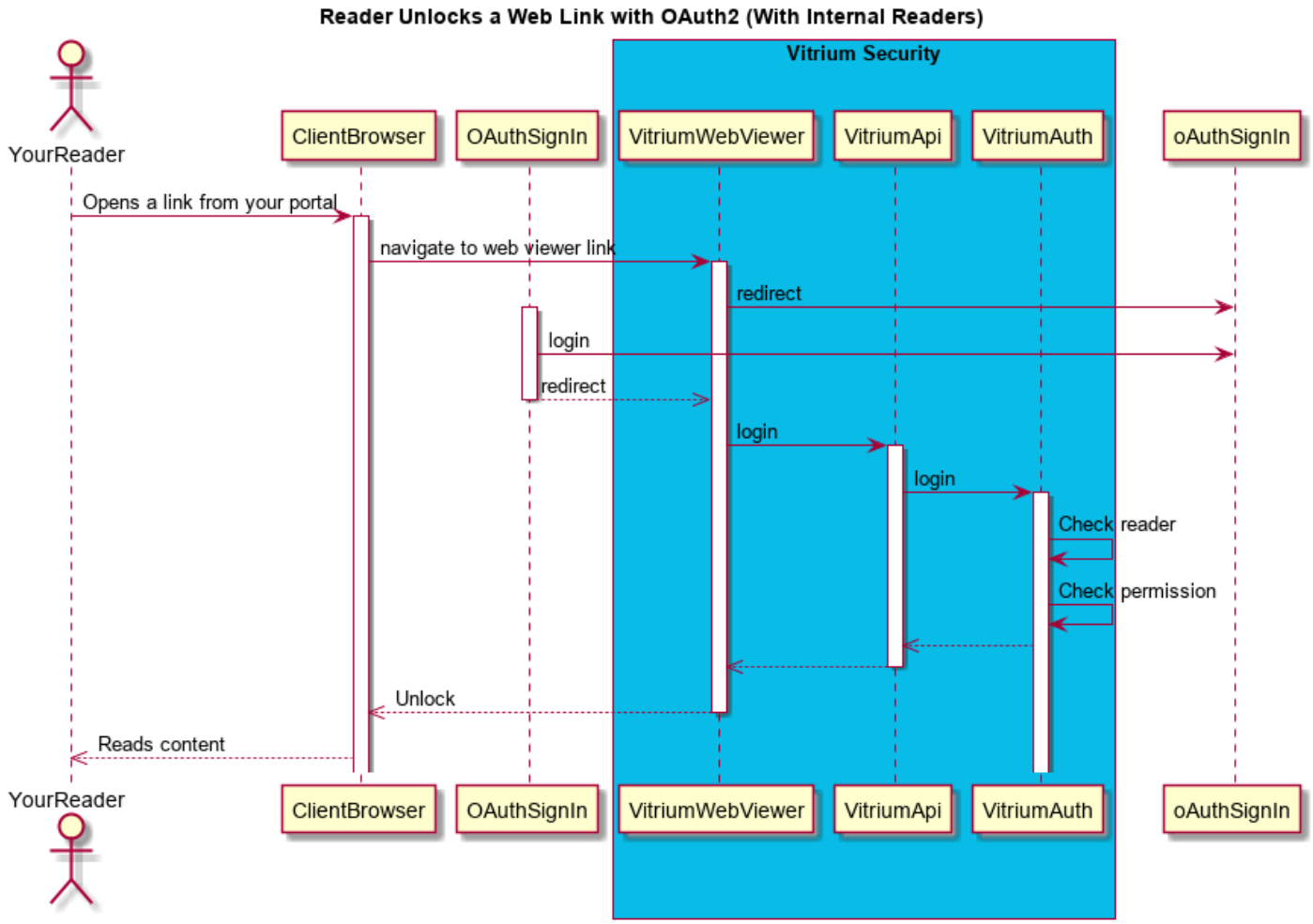


Protected PDF Process (External Auth/EIP):



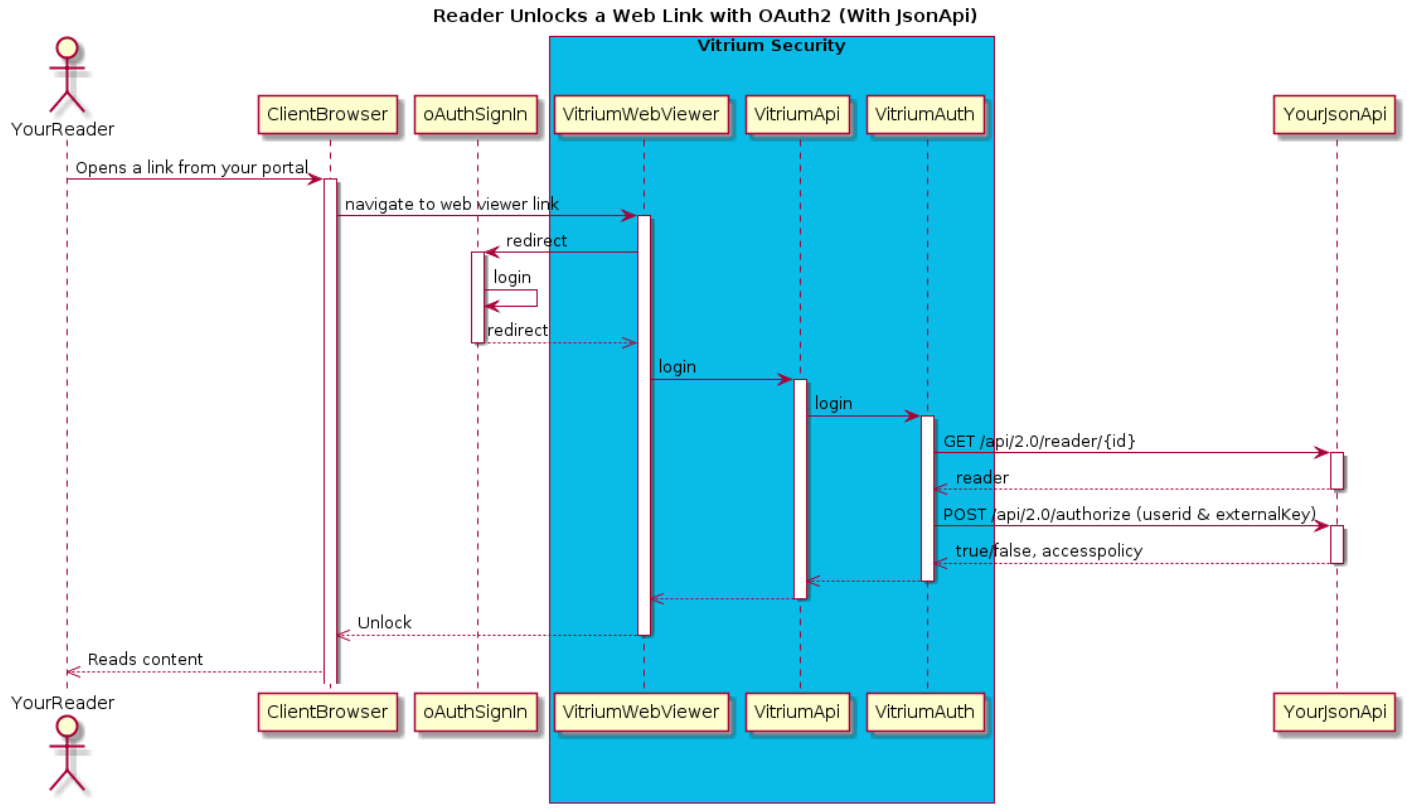
Secured Web Link Process (Vitrium Auth/EIP Using OAuth2):

The following implementation using OAuth2 against Vitrium's Users doesn't require a JSON API implementation.



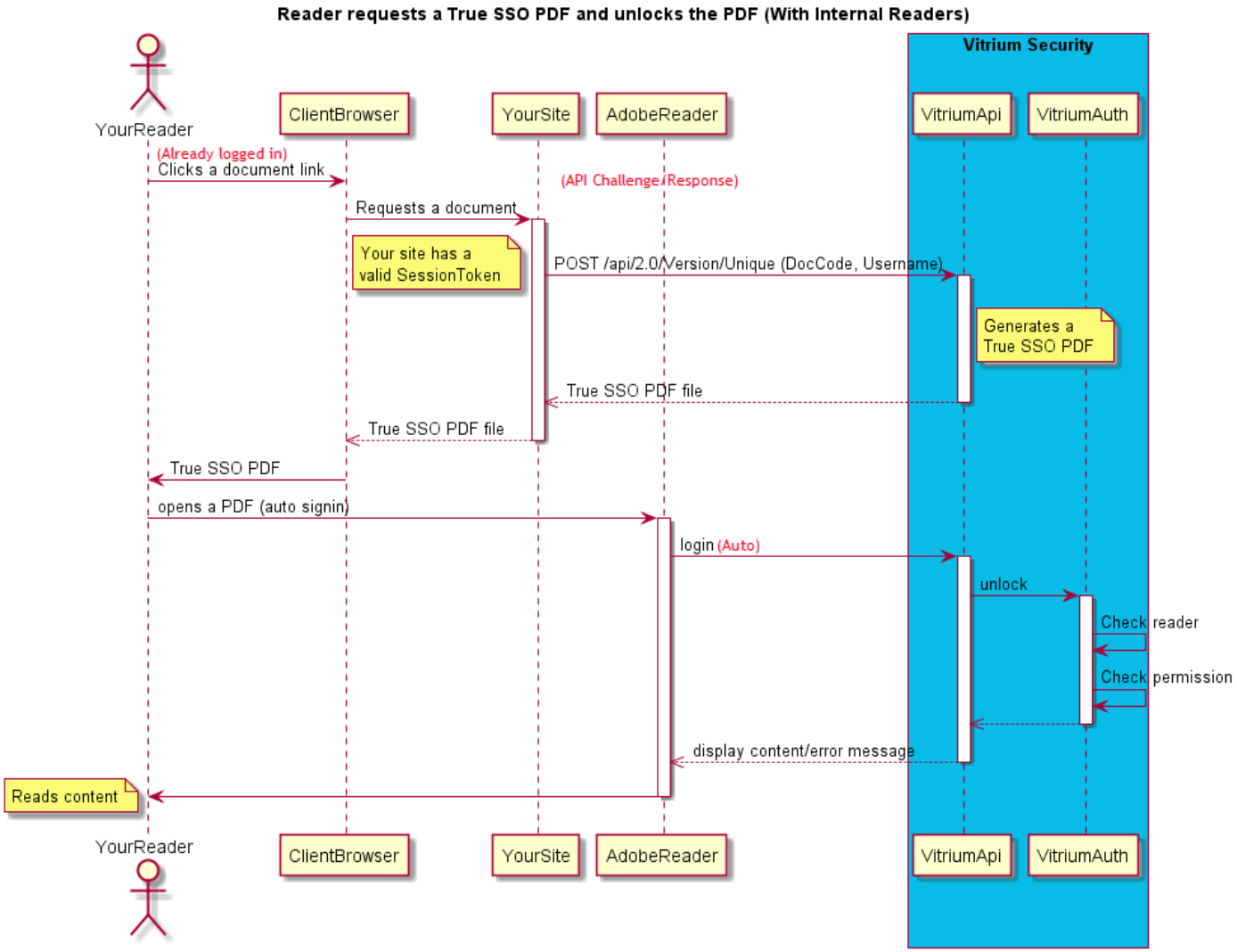
Secured Web Link Process (External Auth/EIP Using OAuth2):

The following is an implementation of External Auth with OAuth2



Protected PDF Process (External Auth/EIP):

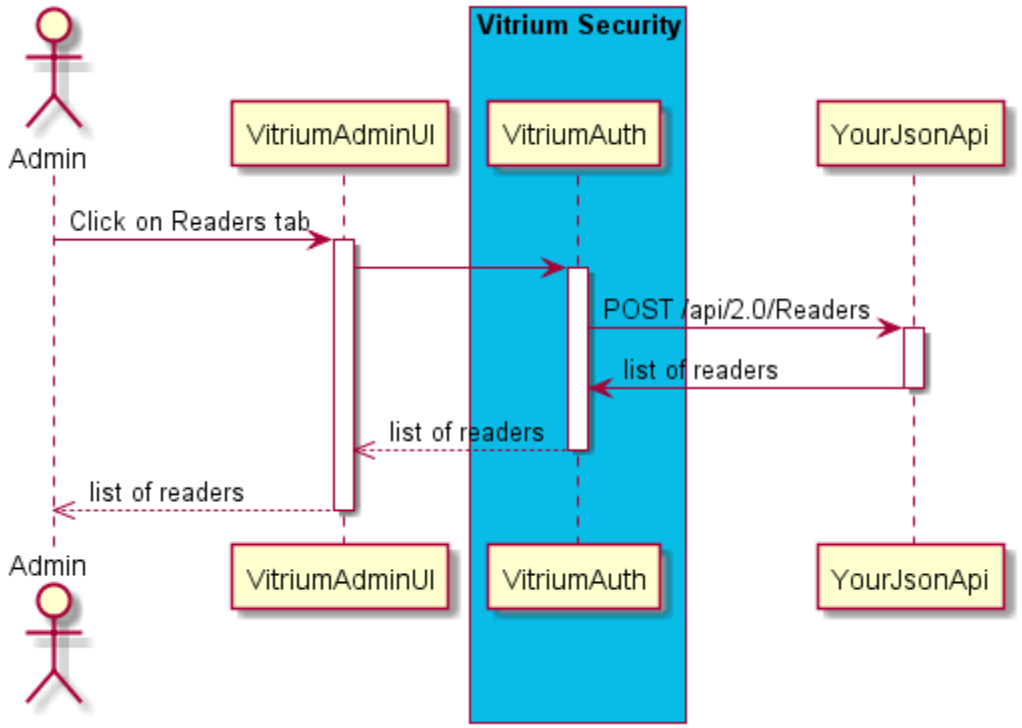
The following implementation using OAuth2 against Vitrium's Users doesn't require a JSON API implementation.



For Viewing Users in Vitrium Admin Interface

When you wish to view all the users or a single user in the Users tab of the Vitrium software, the following API calls can be used:

API Call	Purpose
GET /2.0/readers	Shows all the users in Vitrium's administrative user interface (in the Users tab)
GET /2.0/reader/{Id}	Shows only a single user in Vitrium's administrative user interface



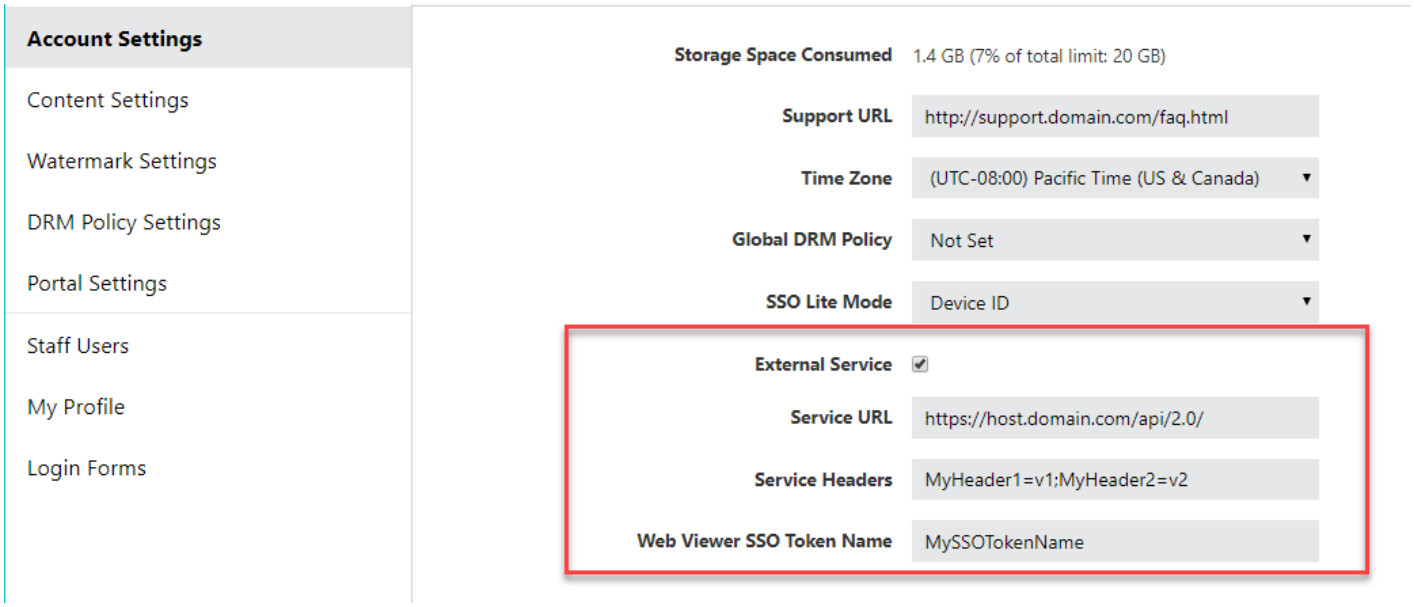
It's important to understand that until you implement the Readers endpoint in your JSON API there will not be and users/readers presented in the Vitrium Admin UI under the Users tab.

Configuring Vitrium Admin User Interface

Enable External Services

Once you have implemented the JSON APIs per this document, you are ready to configure Vitrium's software to talk to your authentication and authorization systems. Here are the steps to do this:

- Log in to your Vitrium Enterprise account:
- Click the **Settings** tab
- In the Global Settings/Account Settings, check the "**External Service**" field to enable it
    - **Service URL:** provide your JSON API Service URL
        - Using a secured site with "https" and an SSL certificate is highly recommended
        - Be sure to suffix your server with "/api/2.0/"
    - **Service Headers:** this is optional if you wish to provide any header key value pairs
        - Separate each header key/value pair with semicolons
        - Separate each key and value with an equal sign
        - Example: key1=value1; key2=value2
    - **Web Viewer SSO Token Name:** this is also optional but required for true SSO



Enable Web Viewer SSO

If implementing True SSO, you will need to setup the Web Viewer SSO functionality. The Web Viewer needs to be configured to read a cookie or the query parameter, and the Vitrium External Service on the account needs to be configured to pass the parameter through to the "Reader by Token" endpoint in the JSON Server.

Web Viewer Configuration

The sso.config file should be placed in your Webviewer repository folder. If this folder location (…\resources\SignIn) doesn't exist, then it should be created:

…\[Account Id]\resources\SignIn\

Examples:

```
C:\Vitrium\Repository\webviewer\19\191c44f9-d9ab-4e0f-b5cd-02be4eb7c305\resources\SignIn\sso.config

C:\Vitrium\API\Repository\webviewer\19\191c44f9-d9ab-4e0f-b5cd-02be4eb7c305\resources\SignIn\sso.config

C:\Vitrium\Docs\Repository\webviewer\19\191c44f9-d9ab-4e0f-b5cd-02be4eb7c305\resources\SignIn\sso.config
```

Tip: Your Account ID can be found under "My Profile" in the Vitrium Admin UI.

SSO.Config File

In the following two examples, a cookie or query parameter named "MySSOTokenName" is used. You may rename the cookie name and/or query parameter string as required.

SSO.Config for Cookie:

```
{ "CookieNames":["MySSOTokenName"],
"RememberMe":true }
```

SSO.Config for Query Parameter:

```
{ "QueryParameters":["MySSOTokenName"],
"RememberMe":true }
```

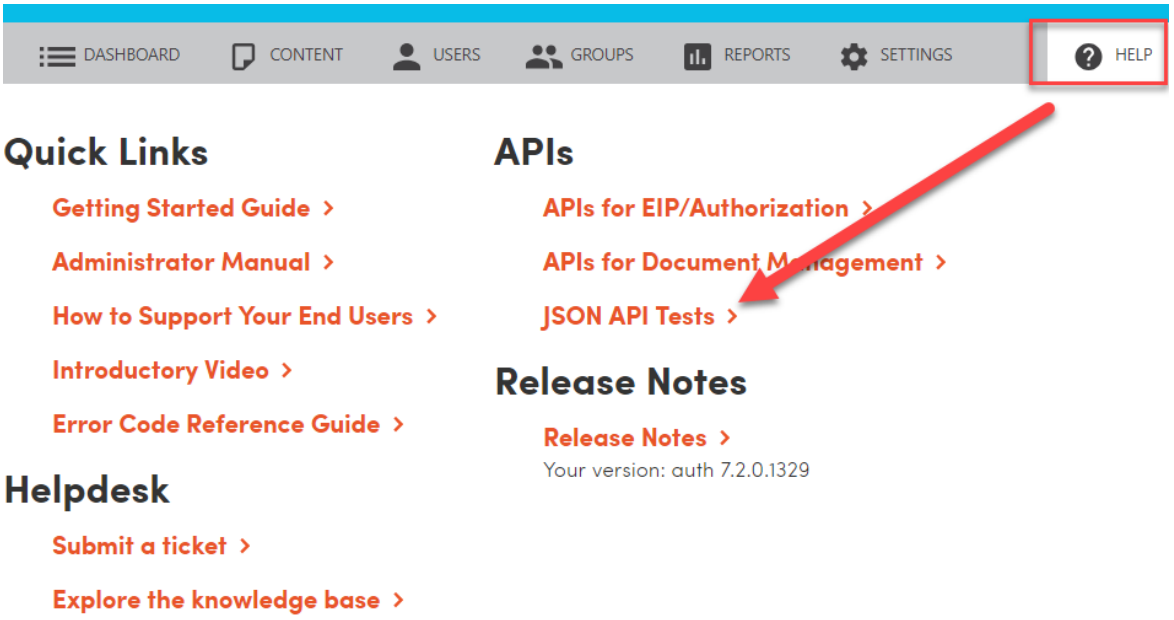
Web Viewer SSO Token Name

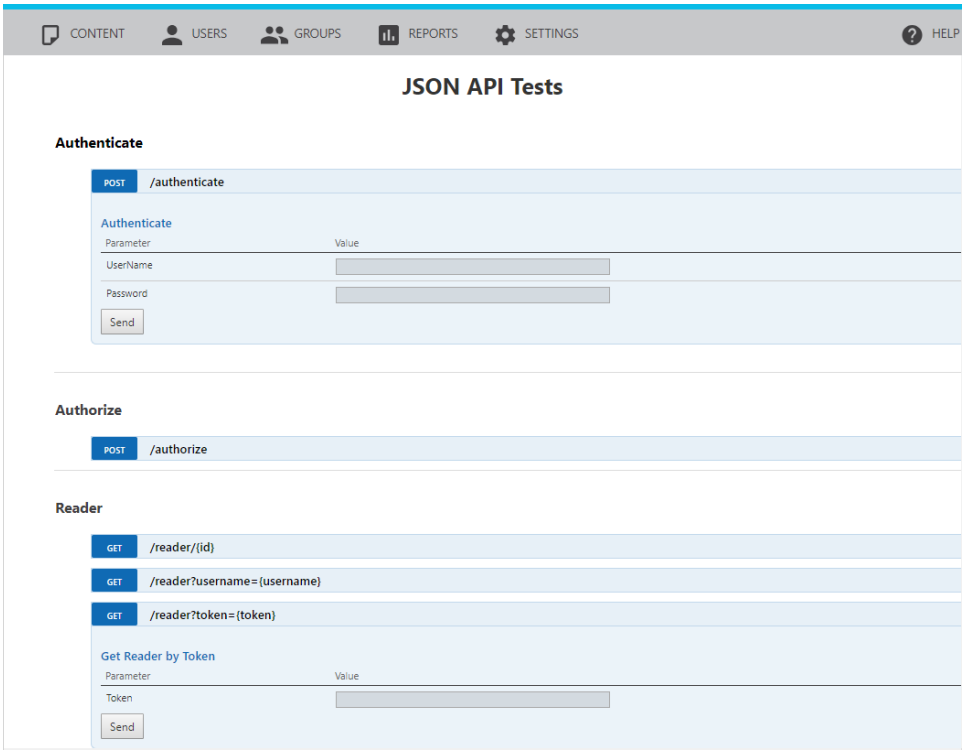
Ensure the token name string is referenced in Global Settings/Account settings. To configure the External Service, navigate to Settings -> Global Settings/Account Settings and check the "External Service" checkbox if it isn't checked already to enable external services. Enter the name of the token (ie MySSOTokenName) under Web Viewer SSO Token Name and click Save Settings.

Testing Your JSON APIs

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

NOTE: The JSON API Tests link will only function if an "External Service" (ie a JSON Server) in the Settings/Global Settings tab is enabled.

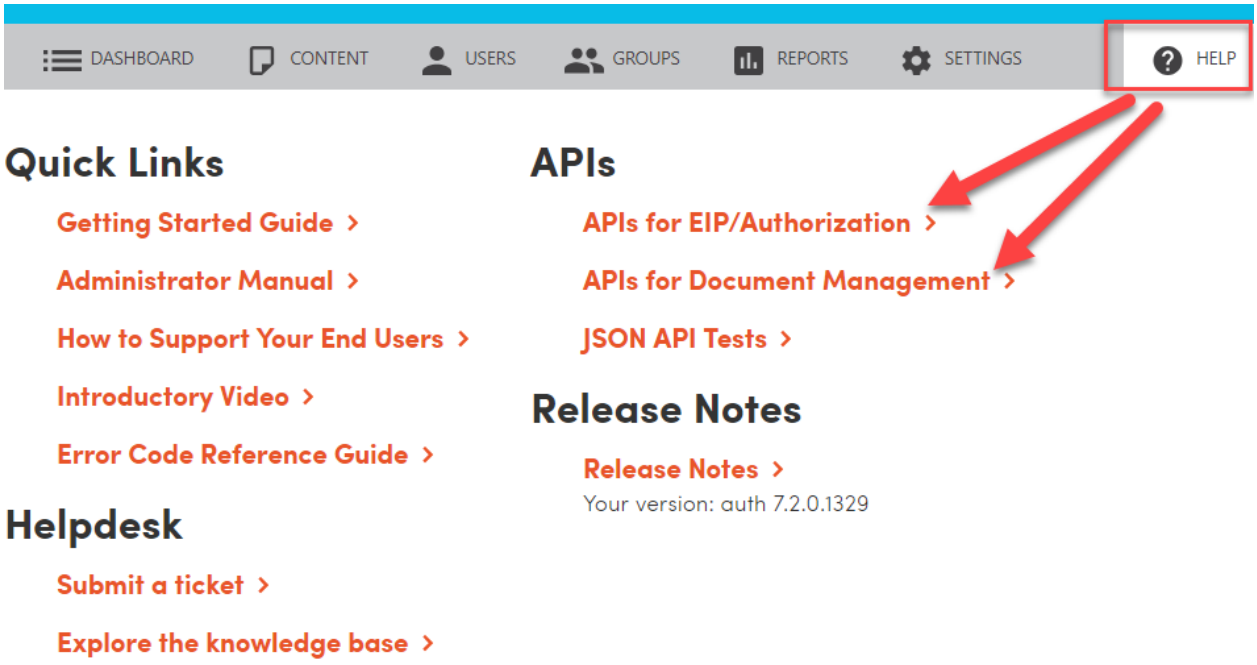


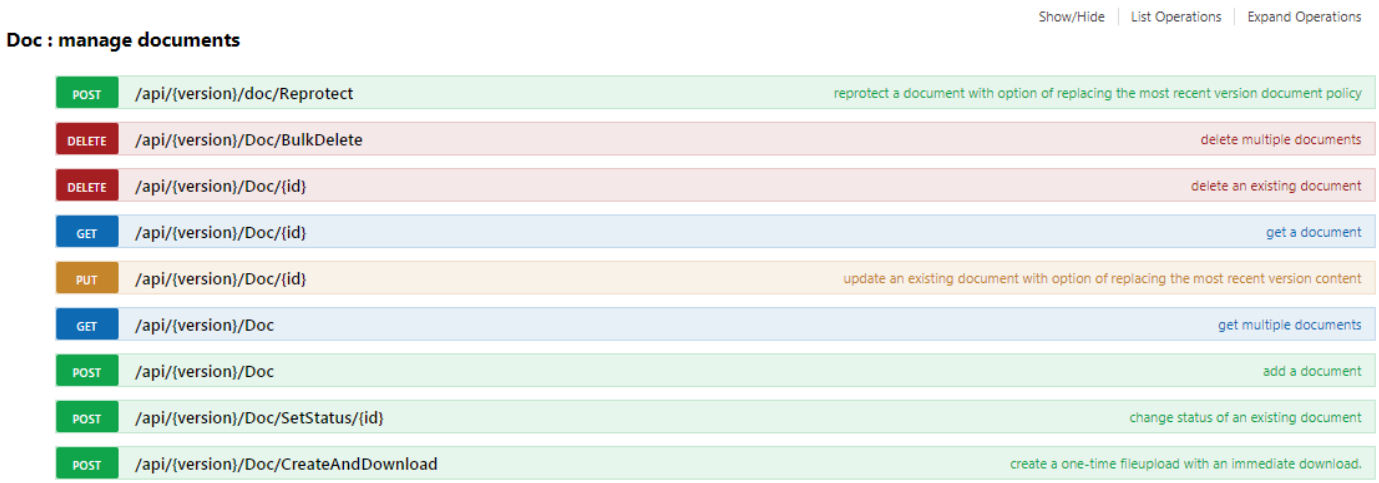


When you access the JSON API Tests page, click on operation that you wish to test and it will expand to reveal the required parameters. Enter test criteria as needed and then click the Send button. This will call that Endpoint with the parameters that you entered. Your Endpoint should return the required response which will be then be displayed.

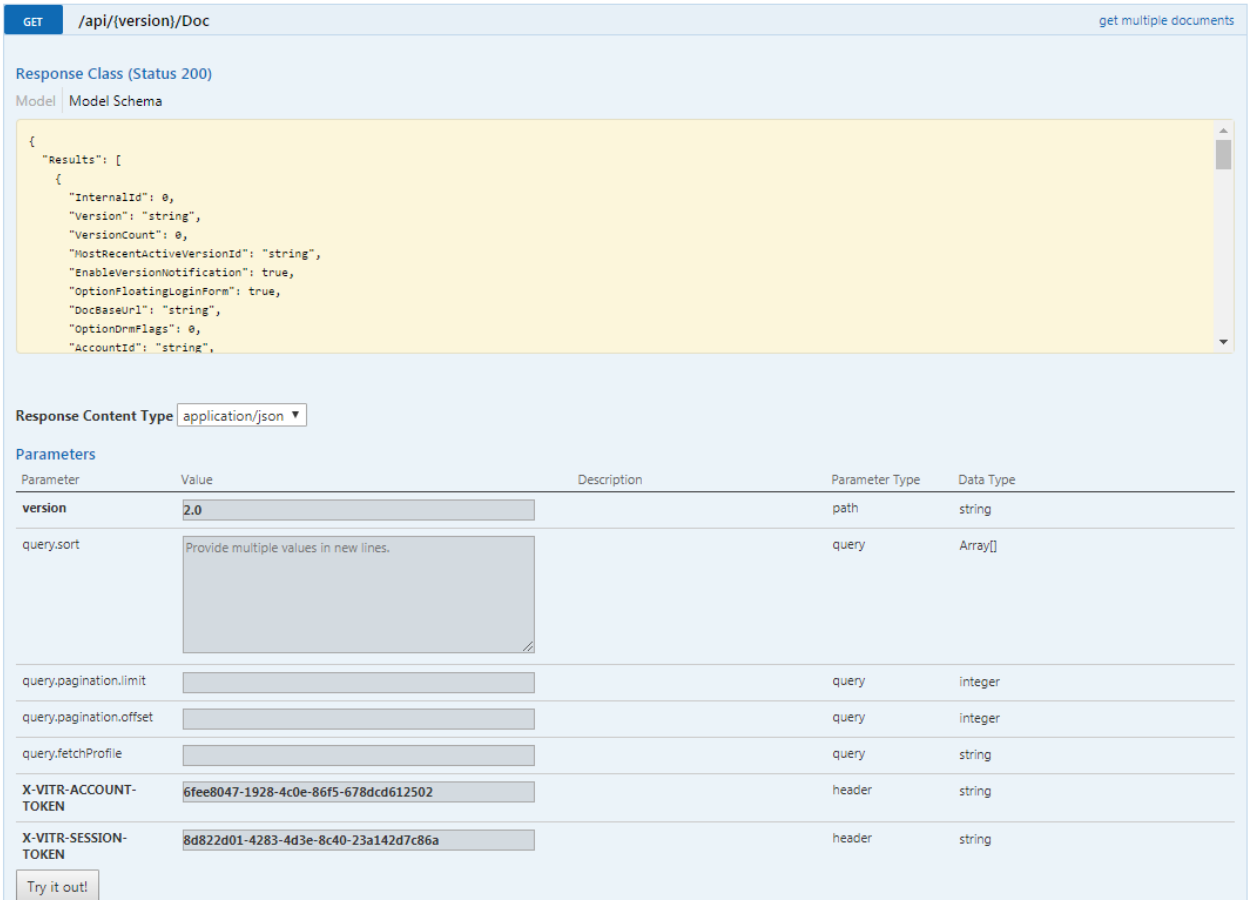
Automating Your Document Workflow

If you wish to automate the document protection and upload process, you can also do this with Vitrium's Document Management APIs. You can access these APIs from the Help tab within your Vitrium Enterprise account:





You can expand on any of the operations and try them out once you enter the parameters.


GET
/api/{version}/Doc
get multiple documents
Response Class (Status 200)
Model
Model Schema
{
"Results": [
{
"InternalId": 0,
"Version": "string",
"VersionCount": 0,
"MostRecentActiveVersionId": "string",
"EnableVersionNotification": true,
"OptionFloatingLoginForm": true,
"DocBaseUrl": "string",
"OptionDrmFlags": 0,
"AccountId": "string",
Response Content Type
application/json
Parameters
Parameter
Value
Description
Parameter Type
Data Type
version
2.0
path
string
query.sort
Provide multiple values in new lines.
query
Array[]
query.pagination.limit
query
integer
query.pagination.offset
query
integer
query.fetchProfile
query
string
X-VITR-ACCOUNT-TOKEN
6fee8047-1928-4c0e-86f5-678dcd612502
header
string
X-VITR-SESSION-TOKEN
8d822d01-4283-4d3e-8c40-23a142d7c86a
header
string
Try it out!