VITRIUM.COM

Vitrium API Guide for Adding Files, Folders, Users, Groups and More

For Vitrium Security v10.16+

Updated: March 20, 2023
Document Revision: 1.02

1.604.677.1500 (direct)
1.866.403.1500 (toll-free)

www.vitrium.com

Table of Contents

Document Revisions

Revision Number	Reason	Date
1.00	Created revision table	October 15, 2018
1.01	Fixed typos	June 18, 2019
1.02	Updated Document API Endpoint Added PHP Code Sample in Appendix A Corrected AES-256 reference to AES-128 Updated links to manuals	March 20, 2023

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:

Documents	Images	Videos[1]
PDF (.pdf) Microsoft Word (.doc / .docx) Microsoft Excel (.xls / .xlsx) Microsoft PowerPoint (.ppt / .pptx) Comma separated values (.csv) Rich text or text files (.rtf, .txt) OpenOffice files (.odt, .ods, .odp)	JPG (.jpg / .jpeg) PNG (.png) GIF (.gif) - *except animated GIFs* TIF (.tif / .tiff) BMP (.bmp)	MOV MP4 WMV AVI FLV MKV

[1] To protect video formats, you will need to have video enabled on your account. All trial accounts will have video enabled. If you are an existing customer and you would like to try video, you will need to contact your Vitrium account manager as there may be an additional charge after a free trial period.

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	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: https://www.vitrium.com/hubfs/support-pdfs/vitrium-api-getting-started-guide.pdf
- Administrator Manual: https://www.vitrium.com/knowledgebase/guides/vitrium-security-admin-manual

About Vitrium APIs

The Vitrium Security APIs are truly RESTful. The API supports true REST-based HTTP methods, i.e.: GET, POST, PUT, DELETE, etc. Vitrium's own application and user interface uses the same APIs so you can be assured that they are comprehensive and mature.

Document & User APIs

There are two sets of APIs available for Vitrium Security Enterprise Edition:

1. **Document APIs** for managing your documents, folders, login forms, watermarks and more. You can try out these APIs from within your Vitrium account at: https://enterprise.protectedpdf.com/secure/docapi
2. **User APIs** for managing your users, groups, DRM policies, and permissions. You can try out these APIs from within your Vitrium account at: https://enterprise.protectedpdf.com/secure/authapi

Implementing the Document APIs

Document API Endpoint

The document APIs supported in a hosted environment are available to Vitrium Enterprise customers. For installed customers, you have complete control over your environment and should use your local API endpoint instead.

Vitrium hosted in USA: https://docs.vitrium.com
Vitrium hosted in Canada: https://docs-ca.vitrium.com
Installed: https://docs.yourdomain.com (or similar)

A complete endpoint is formed by combining the REST verb with the full URI to the resource you are addressing. For example, here is the endpoint you would use in a request to get a specific document using the API:

```
GET [DocumentAPIBase]/api/2.0/doc/{id}
```

To create a complete request, combine the endpoint with the appropriate HTTP headers and your JSON payload.

Routing

The routing is:

```
<HTTP method> <Base URL>/api/2.0/<controller>/<optional entity
id>?param1=value1&param2=value2&...
```

Controllers

The current controllers are as follows:

- Login – login to the API
- Activity – retrieve one or more activity logs
- Doc – manage your documents
- Folder – manage your folders
- Form – manage your forms
- Policy – manage your policies

- Version – managing multiple versions of a document
- Watermark – manage your watermarks

Authentication

At Vitrium, we focus on security. A lot of web services on the Internet don't have very strong security – they simply use an application ID and a secret key that are sent in plaintext (sometimes over SSL). However, Vitrium Security's API authentication requires a 2-way handshake (also known as challenge / response) and your password is only sent in a hashed form based one-time hashing key.

Once you successfully login, a session token will be returned (GUID), which must be sent in all subsequent HTTP request header in the field "X-VITR-SESSION-TOKEN".

API session tokens have an expiry, which is automatically extended after each request. Each response will contain the header "ApiSessionExpiry" that contains the expiry UTC-based timestamp.

Logging In

If you don't already have an account, contact Vitrium sales staff to sign up for a trial account. You will be provided with a username (usually an email address) and a password, which you can login to change. You can use these same credentials to talk to the APIs.

With a POST request like this:

```
POST [DocumentAPIBase]/api/2.0/login/challenge
{
        "ClientNonce": "A8673858-023C-4F8D-805D-FC8D65613A9C"
}
```

Where the ClientNonce is a unique but random GUID that you generate. **Do not reuse** previous Client Nonces. You will get a response like the following:

```
{
      ServerNonce: "2d2d0a7b-c439-413c-a823-0b2acbecd496"
}
```

Remember that **"Token"** is your API session token.

The **clientHash** is: **SHA1**(data = LOWER(clientNonce) + LOWER(serverNonce) + user's plaintext password, key = user's plaintext password). The clientHash should be sent in UPPERCASE hexadecimal format. (see Appendix A on page 12 for detail)

For example: `SHA1("a8673858-023c-4f8d-805d-fc8d65613a9c2d2d0a7b-c439-413c-a823-0b2acbecd496mypassword", "mypassword")` based on above scenario and assuming the user's password is "mypassword" results in hash `D9BB04C0A7F350835E9AE4EED425AEA82EACB31A`

HINT: Use the following online SHA1 conversion tool: http://hash.online-convert.com/sha1-generator

Then, with a follow up POST request like this:

```
POST [DocumentAPIBase]/api/2.0/login/response
{
        "ClientNonce" : "a8673858-023c-4f8d-805d-fc8d65613a9c",
        "UserName" : "staffuser@yourdomain.com",
        "ClientHash" : "D9BB04C0A7F350835E9AE4EED425AEA82EACB31A"
}
```

You will get a response like the following:

```
{
      ServerHash: "BE2808CD258A8A3A187FADA5AF9B33DE56DFA174"
      ApiSession:
      {
            Token: "210e0825-09f4-40ec-83ed-731897bb6ef0"
            Expiry: "2014-08-27T23:33:59.0451143Z"
      }
      Accounts: [1]
      0: {
            Id: "07f40b2d-7d0d-4ae0-aead-8dd33b4e8fb4"
            Name: "Your account name"
            WebViewerDocEncryptionEnabled: true
            AllowUserDefinedWebViewerEncryptionPassword: false
            AccountType: "enterprise"
            AnalyticsEnabled: true
      }
      User:
      {
            Id: "5a2bc179-df0a-4fe2-8c63-dfdb40441a4d"
            FirstName: "Staff"
            LastName: "User"
            Email: "staffuser@yourdomain.com"
            IsFirstLogin: false
            HasDocument: true
      }
}
```

Creating a Folder

Once you're logged in, now you are ready to talk to the secure APIs, but be sure to include both your API session token and account ID in the header:

With a GET request like this:

```
GET [DocumentAPIBase]/api/2.0/folder/
```

with the following header:

```
X-VITR-SESSION-TOKEN: 210e0825-09f4-40ec-83ed-731897bb6ef0
X-VITR-ACCOUNT-TOKEN: 0?f40b2d-7d0d-4ae0-aead-8dd33b4e8fb4
```

You will get a response like this:

```
{
      Results: [2]
      0: {
            Id: "8c6124dc-e204-46b3-b7b8-8cba5dfffd06"
            ExternalKey: null
            Name: "Private Folder - Staff User"
            ParentFolderId: null
            HasSubFolder: false
            CustomField: null
      }
      1: {
            Id: "d3f9c965-73f7-4852-84c4-dfe3cbb10579"
            …
      }
      TotalRecords: 2
}
```

And to get a specific document, use this:

```
GET [DocumentAPIBase]/api/2.0/version/<id>
```

Uploading a File for Protection

Once you've set up your folder structure, you can now upload files for protection.

With a POST request like this:

```
Multipart/form-data POST [DocumentAPIBase]/api/2.0/doc
{
     rawfile: your unprotected file
     settings: your new document metadata in JSON format
}
```

Where your settings JSON file would look something like this:

```
{
    "FolderId": "8c6124dc-e204-46b3-b7b8-8cba5dfffd06",
    "FileName": "TestByStaffUser",
    "Title": " TestByStaffUser ",
    "PolicyId":"58ef4f2e-20f1-4f5a-9eeb-dc0371606a56",
    "Status": "Active",
}
```

You will get a response like the following:

```
{
      DocumentId: "ef02b71d-b6aa-4b7d-811f-ba15069a05e7"
      Document:
      {
            FolderId: "8c6124dc-e204-46b3-b7b8-8cba5dfffd06"
            FolderIdPath: "8c6124dc-e204-46b3-b7b8-8cba5dfffd06"
            FileName: "TestByStaffUser"
            Title: " TestByStaffUser "
            PolicyId: "58ef4f2e-20f1-4f5a-9eeb-dc0371606a56"
            Id: "ef02b71d-b6aa-4b7d-811f-ba15069a05e7"
            CreatedOn: "2014-08-28T22:45:44.2190628Z"
      …
}
```

Generating a True SSO PDF Document from an Existing Document

With a POST request like this:

```
POST [DocumentAPIBase]/api/2.0/Version/Unique
{
      "DocCode": "0000-1836-1C869-0001E1D9",
      "DocPolicyOverride": {
            "AcroJsGosBehaviourType": "PromptAndCloseDocument",
            "PrintType": "HighResolution"
      },
      "AccessPolicyOverride": {
            "ComputersMax": 2,
            "IpAddressesMax": null,
            "IgnoredIpAddresses": null,
            "OfflineDurationinDays": 7,
            "DocumentLimit": 1,
            "OpenLimit": null,
            "RelativeExpiryInDays": null,
            "ExpiryInMins": 52560000
      },
      "UniqueDocCopyId": "60efba43-8a30-4558-8c62-348fa21a8bf6",
      "Username": "staffuser@yourdomain.com"
}
```

You'll get the file back in the response.

Implementing the User APIs

The optional User API allows you to manage users (also known as readers), groups, permissions and DRM policies using Vitrium's software instead of your own system.

NOTE: these APIs cannot be used when Vitrium Security is configured to use an external service either via an AuthExtension module (DLL) or when you implement JSON APIs.

User API Endpoint

The User APIs are supported in a production environment for Vitrium's hosted customers. For installed customers, you have complete control over your environment and should use your local API endpoint instead.

Vitrium hosted in USA: https://security.vitrium.com
Vitrium hosted in Canada: https://security-ca.vitrium.com
Installed: https://admin.yourdomain.com (or similar)

A complete endpoint is formed by combining the REST verb with the full URI to the resource you are addressing.

Routing

The routing is:

```
<HTTP method> <Base URL>/api/2.0/<controller>/<optional entity id>?param1=value1&param2=value2&...
```

The API supports true REST-based HTTP methods, i.e.: GET, POST, PUT, DELETE, etc.

Controllers

The current controllers are as follows:

- AccessPolicy – manage your DRM policies
- Activity – retrieve one or more activity logs
- Group – manage your groups
- Permission – manage your permissions
- Reader – manage your users (readers)
- Settings – manage your account settings
- UniqueDoc – register unique documents generated on-demand
- Unlock – perform a remote unlock

Authentication

Once you've logged into the Document API, you don't need to login to the User API. You can simply reuse your document API session.

As with the Document APIs, be sure to include the same API session token and account ID in the header, e.g.:

```
GET [UserAPIBase]/api/2.0/reader/
```

with the same header:

```
X-VITR-SESSION-TOKEN: 210e0825-09f4-40ec-83ed-731897bb6ef0
X-VITR-ACCOUNT-TOKEN: 07f40b2d-7d0d-4ae0-aead-8dd33b4e8fb4
```

You will get a response like:


```
{
	Results: [1]
	0:
	{
		Id: "af216d96-3a4f-4a19-8eba-1b3262fb4dde"
		Username: "staffuser@yourdomain.com"
		Password: "12345"
		PasswordEncrypted: true
		IsActive: true
		Notes: ""
		Policy:
		{
			…
		}
		ExternalKey: ""
		CustomField: ""
		CreatedOn: "2016-10-18T21:10:46.31"
	}
	TotalRecords: 1
}
```

Appendix A

PHP Code Sample

```
<?php

function createGuid() {

  if (function_exists('com_create_guid'))
      return trim(com_create_guid(), '{}');

  return sprintf('%04X%04X-%04X-%04X-%04X-%04X%04X%04X', mt_rand(0, 65535), mt_rand(0,
65535), mt_rand(0, 65535), mt_rand(16384, 20479), mt_rand(32768, 49151), mt_rand(0, 65535),
mt_rand(0, 65535), mt_rand(0, 65535));

}

$guid = createGuid();
$serverNonce = SEVERNONCE
$password    = ACCOUNTPASSWORD

$clientHash  = strtoupper(hash_hmac('sha1',
strtolower($guid).strtolower($serverNonce).$password, $password));

echo $clientHash;
```

Above example returns ClientHash, make sure to modify SERVERNONCE and ACCOUNTPASSWORD accordingly.