BLACKDUCK

BY SYNOPSYS®

Getting Started with the SDK

Version 2019.10.3

This edition of the *Getting Started with the SDK* refers to version 2019.10.3 of Black Duck.

This document created or updated on Thursday, February 6, 2020.

Please send your comments and suggestions to:

Synopsys
800 District Avenue, Suite 201
Burlington, MA 01803-5061 USA

Contents

Black Duck documentation

The documentation for Black Duck consists of online help and these documents:

Title	File	Description
Release Notes	release_notes.pdf	Contains information about the new and improved features, resolved issues, and known issues in the current and previous releases.
Installing Black Duck using Docker Compose	install_compose.pdf	Contains information about installing and upgrading Black Duck using Docker Compose.
Installing Black Duck using Docker Swarm	install_swarm.pdf	Contains information about installing and upgrading Black Duck using Docker Swarm.
Installing Black Duck using Kubernetes	install_kubernetes.pdf	Contains information about installing and upgrading Black Duck using Kubernetes.
Installing Black Duck using OpenShift	install_openshift.pdf	Contains information about installing and upgrading Black Duck using OpenShift.
Getting Started	getting_started.pdf	Provides first-time users with information on using Black Duck.
Scanning Best Practices	scanning_best_practices.pdf	Provides best practices for scanning.
Getting Started with the SDK	getting_started_sdk.pdf	Contains overview information and a sample use case.

Title	File	Description
Report Database	report_db.pdf	Contains information on using the report database.
User Guide	user_guide.pdf	Contains information on using Black Duck's UI.

Black Duck integration documentation can be found on Confluence.

Customer support

If you have any problems with the software or the documentation, please contact Synopsys Customer Support.

You can contact Synopsys Support in several ways:

- Online: https://www.synopsys.com/software-integrity/support.html
- Email: software-integrity-support@synopsys.com
- Phone: See the Contact Us section at the bottom of our support page to find your local phone number.

Another convenient resource available at all times is the online customer portal.

Synopsys Software Integrity Community

The Synopsys Software Integrity Community is our primary online resource for customer support, solutions, and information. The Community allows users to quickly and easily open support cases and monitor progress, learn important product information, search a knowledgebase, and gain insights from other Software Integrity Group (SIG) customers. The many features included in the Community center around the following collaborative actions:

- Connect - Open support cases and monitor their progress, as well as, monitor issues that require Engineering or Product Management assistance
- Learn - Insights and best practices from other SIG product users to allow you to learn valuable lessons from a diverse group of industry leading companies. In addition, the Customer Hub puts all the latest product news and updates from Synopsys at your fingertips, helping you to better utilize our products and services to maximize the value of open source within your organization.
- Solve - Quickly and easily get the answers you're seeking with the access to rich content and product knowledge from SIG experts and our Knowledgebase.
- Share - Collaborate and connect with Software Integrity Group staff and other customers to crowdsource solutions and share your thoughts on product direction.

Access the Customer Success Community. If you do not have an account or have trouble accessing the system, click here to get started, or send an email to community.manager@synopsys.com.

Training

Synopsys Software Integrity, Customer Education (SIG Edu) is a one-stop resource for all your Black Duck education needs. It provides you with 24x7 access to online training courses and how-to videos.

New videos and courses are added monthly.

At Synopsys Software Integrity, Customer Education (SIG Edu), you can:

- Learn at your own pace.
- Review courses as often as you wish.
- Take assessments to test your skills.
- Print certificates of completion to showcase your accomplishments.

Learn more at https://community.synopsys.com/s/education.

Chapter 1: About the Black Duck SDK

Black Duck REST APIs provide a standard interface for interacting with Black Duck.

REST APIs provide access to resources (data entities) via URI paths. To use a REST API, your application makes a HTTP request and parses the response. The methods used by your application are the standard HTTP methods of GET, POST, PUT and DELETE. REST APIs operate over HTTP(S) making it easy to use with any programming language or framework. The input and output format for the majority of Black Duck REST APIs is JSON.

Important: Black Duck does not support REST APIs that begin with "v1" or "internal".

Viewing and Testing the REST APIs

Black Duck provides online access and documentation to the REST servers and individual REST APIs. After logging in to Black Duck, you can enter the following url to access this information:

Access the Black Duck API documentation at https://<hub-server>/api-doc/public.html

Each supported REST API is documented with the required input parameters and expected response.

Media types

The API uses custom media types, which enable you to choose the format of the data that you receive. Requesting a media type enables a stable, backwards-compatible return.
To request a media type, you add it as an Accept header when you make a request, and as a Content-Type header when providing a request body. Responses contain a Content-Type header that describes the format.

Media types are documented with individual endpoints, but are grouped by general categories of resources. Black Duck supports the following media types:

```
application/vnd.blackducksoftware.admin-4+json
application/vnd.blackducksoftware.bill-of-materials-4+json
application/vnd.blackducksoftware.bill-of-materials-5+json
application/vnd.blackducksoftware.bill-of-materials-6+json
application/vnd.blackducksoftware.component-detail-4+json
application/vnd.blackducksoftware.component-detail-5+json
application/vnd.blackducksoftware.journal-4+json
application/vnd.blackducksoftware.notification-4+json
application/vnd.blackducksoftware.policy-4+json
application/vnd.blackducksoftware.policy-5+json
application/vnd.blackducksoftware.project-detail-4+json
application/vnd.blackducksoftware.project-detail-5+json
```

```
application/vnd.blackducksoftware.report-4+json
application/vnd.blackducksoftware.scan-4+json
application/vnd.blackducksoftware.status-4+json
application/vnd.blackducksoftware.user-4+json
application/vnd.blackducksoftware.vulnerability-4+json
```

Here's an example that uses a Content-Type header and an Accept header:

```
PUT /api/usergroups/{userGroupId}
Content-Type: application/vnd.blackducksoftware.user-4+json
Accept: application/vnd.blackducksoftware.user-4+json
```

Black Duck supports the following internal media types for exclusive use by the user interface:

```
application/vnd.blackducksoftware.internal-1+json
application/vnd.blackducksoftware.summary-1+json
```

Authentication

Black Duck enables you to generate one or more tokens for accessing Black Duck APIs.
With access tokens, if a security breach occurs, the user’s credentials (which might be their SSO or LDAP credentials) are not directly compromised.

To access Black Duck API, you must authenticate by doing the following steps:

1. Generate an API token in Black Duck by using the Black Duck UI to navigate to your Profile page to generate API token.
2. Pass the API token in an HTTP POST to `/api/tokens/authenticate` to generate a Bearer token, which you use for authorization.
3. Pass the bearer token in the authorization header of you API requests to get data from your Black Duck instance.

Generating an API token

Log in to your Black Duck instance and generate an API token.

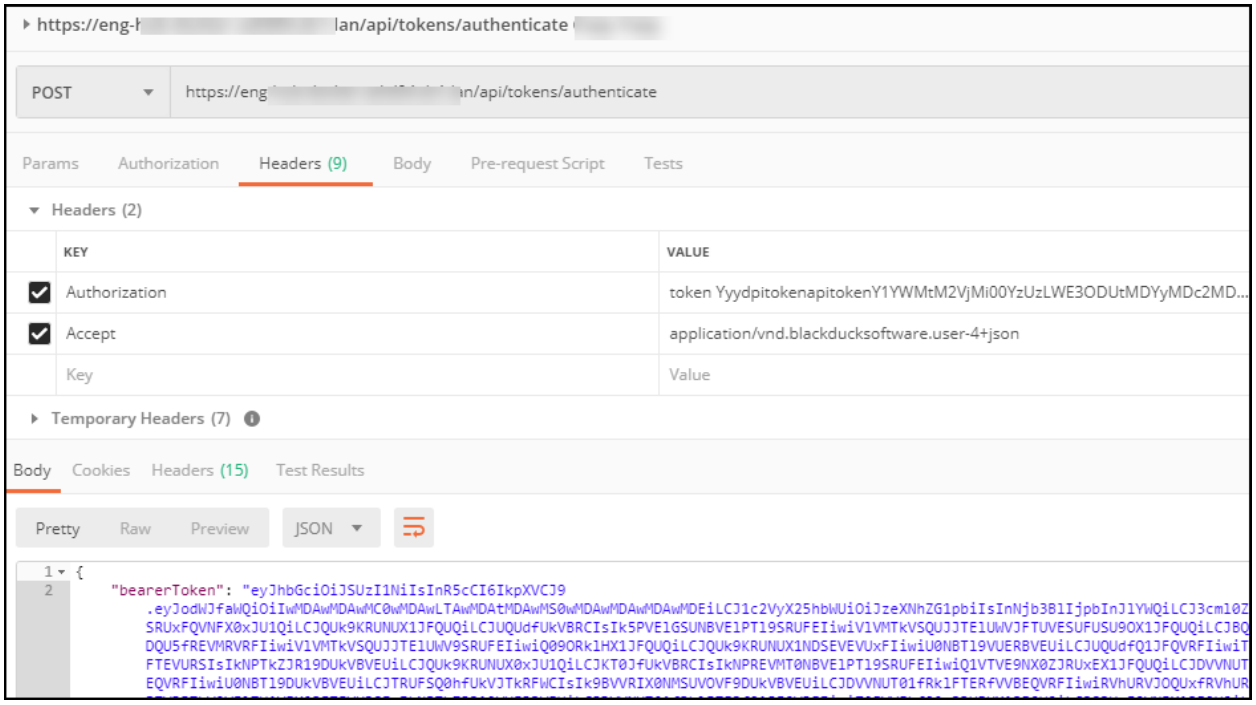
To generate an API token

1. On the top navigation bar, click **System** > **My Profile**.
2. In the **User Access Token** section, type a name in the **Name** field.
3. **Optional:** In the **Description** field, type a description or definition.
4. Select **Read Access** and/or **Write Access**.
5. Click **Generate**.
   The API token displays in a pop-up window. For security reasons, this is the only time your user API token displays. Please save this token. If the token is lost, you must regenerate it.

Using API token

To use an API token, make an HTTP POST to `/api/tokens/authenticate` with the following header:
`Authorization: token <API token>`

This produces a response with a bearer token.



▸ https://eng-[redacted]lan/api/tokens/authenticate [redacted]

POST ▾ https://eng[redacted]n/api/tokens/authenticate

Params Authorization **Headers (9)** Body Pre-request Script Tests

▾ Headers (2)

	KEY	VALUE
✓	Authorization	token YyydpitokenapitokenY1YWMtM2VjMi00YzUzLWE3ODUtMDYyMDc2MD...
✓	Accept	application/vnd.blackducksoftware.user-4+json
	Key	Value

▸ Temporary Headers (7)

Body Cookies Headers (15) Test Results

Pretty Raw Preview JSON ▾

```
{
    "bearerToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJodWJfaWQiOiIwMDAwMDAwMC0wMDAwLTAwMDAtMDAwMS0wMDAwMDAwMDAwMDEiLCJ1c2VyX25hbWUiOiJzeXNhZG1pbiIsInNjb3BlIjpbInJ1YWQiLCJ3cml0Z
        SRUxFQVNFX0xJU1QiLCJQUk9KRUNUX1JFQUQiLCJUQUdfUkVBRCIsIk5PVE1GSUNBVE1PT19SRUFEIiwiV1VMTkVSQUJJTE1UWVJFTUVESUFUSU9OX1JFQUQiLCJBQ
        DQU5fREVMRVRFIiwiV1VMTkVSQUJJTE1UWV9SRUFEIiwiQ09ORk1HX1JFQUQiLCJQUk9KRUNUX1NDSEVEVUxFIiwiU0NBT19VUERBVEUiLCJUQUdfQ1JFQVRFIiwiT
        FTEVURSIsIkNPTkZJR19DUkVBVEUiLCJQUk9KRUNUX0xJU1QiLCJKT0JfUkVBRCIsIkNPREVMT0NBVE1PT19SRUFEIiwiQ1VTVE9NX0ZJRUxEX1JFQUQiLCJDVVNUT
        EQVRFIiwiU0NBT19DUkVBVEUiLCJTRUFSQ0hfUkVJTkRFWCIsIk9BVVRIX0NMSUVOVF9DUkVBVEUiLCJDVVNUT01fRk1FTERfVVBEQVRFIiwiRVhURVJOQUxfRVhUR
```

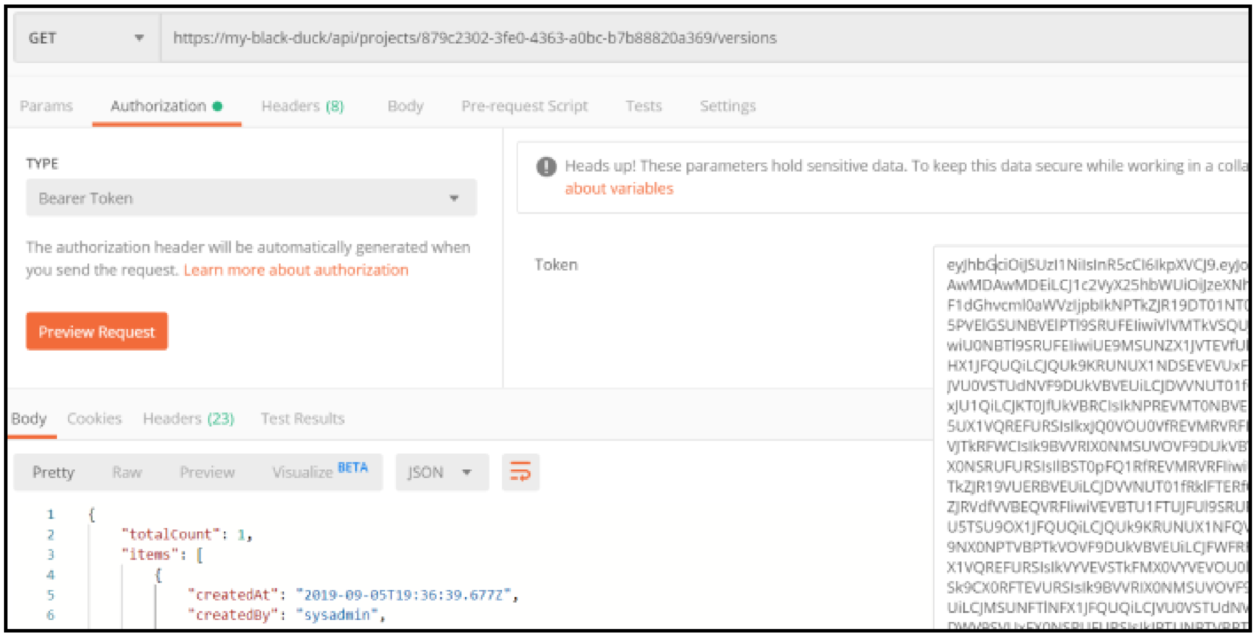
The following example shows a cURL POST request in Postman with the API token (Authorization token) from Black Duck to generate the Bearer token.

```
curl -X POST \
 -H 'Accept: application/vnd.blackducksoftware.user-4+json' \
 -H 'Authorization: token

YjU4ODkyNmItODI1Ny00NjRkLWJlNDEtMWE0MzE3MmFiODgwOmMxMzNkMzk4LWY5NzItNDkwNS05N
TdjOQ=='
```

Using the bearer token in API calls

The following example in the Postman client shows a GET request using the bearer token that was generated from the API token.


GET https://my-black-duck/api/projects/879c2302-3fe0-4363-a0bc-b7b88820a369/versions

Params Authorization ● Headers (8) Body Pre-request Script Tests Settings

TYPE

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a colla

about variables

Token

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJo
AwMDAwMDEiLCJ1c2VyX25hbWUiOiJzeXNh
F1dGhvcml0aWVzIjpbIkNPTkZJR19DT01NT
5PVElGSUNBVElPTl9SRUFEIiwiVlVMTkVSQU
wiU0NBTl9SRUFEIiwiUE9MSUNZX1JVTEVfUl
HX1JFQUQiLCJQUk9KRUNUX1NDSEVEVUxF
JVU0VSTUdNVF9DUkVBVEUiLCJDVVNUT01f
xJU1QiLCJKT0JfUkVBRCIsIkNPREVMT0NBVE
5UX1VQREFURSIsIkxJQ0VOU0VfREVMRVRF
VJTkRFWCIsIk9BVVRIX0NMSUVOVF9DUkVB
X0NSRUFURSIsIlBST0pFQ1RfREVMRVRFIiwi
TkZJR19VUERBVEUiLCJDVVNUT01fRkIFTERf
ZJRVdfVVBEQVRFIiwiVEVBTU1FTUJFUl9SRU
U5TSU9OX1JFQUQiLCJQUk9KRUNUX1NFQV
9NX0NPTVBPTkVOVF9DUkVBVEUiLCJFWFR
X1VQREFURSIsIkVYVEVSTkFMX0VYVEVOU0
Sk9CX0RFTEVURSIsIk9BVVRIX0NMSUVOVF
UiLCJMSUNFTlNFX1JFQUQiLCJVU0VSTUdN

Body Cookies Headers (23) Test Results

Pretty Raw Preview Visualize BETA JSON

```
{
    "totalCount": 1,
    "items": [
        {
            "createdAt": "2019-09-05T19:36:39.677Z",
            "createdBy": "sysadmin",
```

API requests using authorization-based authentication

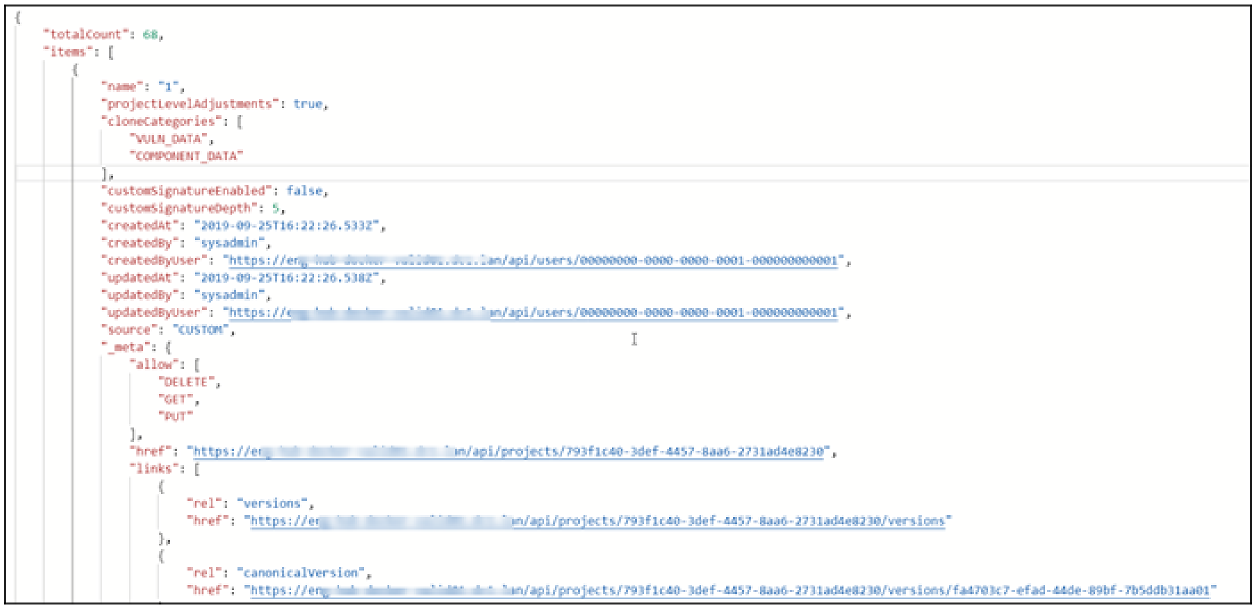
Note how the token provided is sent back in the subsequent request within the 'Authorization' header with 'Bearer' authentication scheme
Additionally, no CSRF, (as described here) is included since it is not required when using the 'Authorization' header to pass the token.

```
curl -v -H "Authorization: Bearer <token>" -X GET
"https://<hostname>/api/components/b2d3d36c-15d3-4e1a-bef5-
8e907f107048/versions/35f6d76e-e4dd-4b87-bae5-133da093dcd2"
```

The following example shows a cURL request for projects:

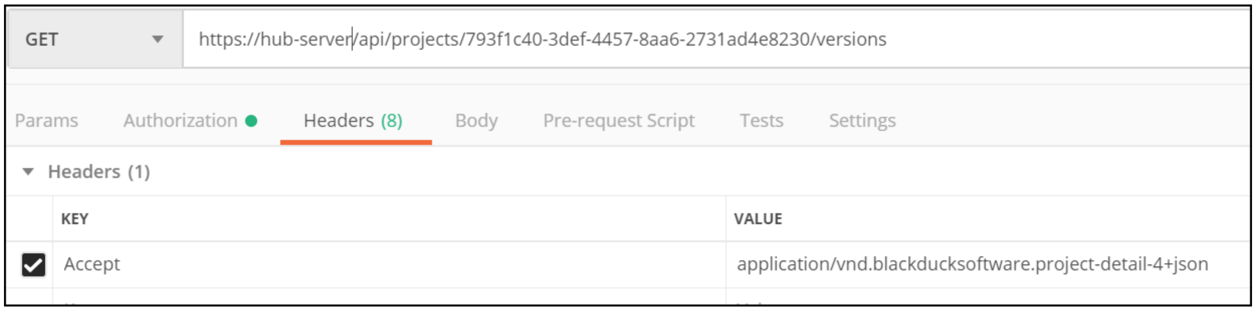
```
curl -X GET \
https://<hub-server>/api/projects/ \
-H 'Accept: application/vnd.blackducksoftware.project-detail-4+json' \
-H 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJodWJfaWQiOiIwMDAwMDAwMC0wMDAwLTAwMDAt
MDAwMS0wMDAwMDAwMDAwMDEiLCJ1c2VyX25hbW
UiOiJzeXNhZG1...
```

The following image show a partial output from the cURL request.



```
{
    "totalCount": 68,
    "items": [
        {
            "name": "1",
            "projectLevelAdjustments": true,
            "cloneCategories": [
                "VULN_DATA",
                "COMPONENT_DATA"
            ],
            "customSignatureEnabled": false,
            "customSignatureDepth": 5,
            "createdAt": "2019-09-25T16:22:26.533Z",
            "createdBy": "sysadmin",
            "createdByUser": "https://en[redacted]an/api/users/00000000-0000-0000-0001-000000000001",
            "updatedAt": "2019-09-25T16:22:26.538Z",
            "updatedBy": "sysadmin",
            "updatedByUser": "https://e[redacted]an/api/users/00000000-0000-0000-0001-000000000001",
            "source": "CUSTOM",
            "_meta": {
                "allow": [
                    "DELETE",
                    "GET",
                    "PUT"
                ],
                "href": "https://er[redacted]an/api/projects/793f1c40-3def-4457-8aa6-2731ad4e8230",
                "links": [
                    {
                        "rel": "versions",
                        "href": "https://er[redacted]an/api/projects/793f1c40-3def-4457-8aa6-2731ad4e8230/versions"
                    },
                    {
                        "rel": "canonicalVersion",
                        "href": "https://en[redacted]an/api/projects/793f1c40-3def-4457-8aa6-2731ad4e8230/versions/fa4703c7-efad-44de-89bf-7b5ddb31aa01"
```

The following example from Postman shows a request for versions of project 793f1c40-3def-4457-8aa6-2731ad4e8230:



GET https://hub-server/api/projects/793f1c40-3def-4457-8aa6-2731ad4e8230/versions

Headers (1)

	KEY	VALUE
✓	Accept	application/vnd.blackducksoftware.project-detail-4+json

Black Duck APIs are backward compatible within a Black Duck media type (excluding application/json).

To avoid issues, when sending or receiving data, specify the Black Duck media type in the Accept header.

Chapter 2: Example of using the Black Duck SDK

Use Case: I'm interested in using the REST APIs to find risk counts for my project 'Application1' version '1.0'.

Step 1: Authentication

You must first authenticate with the Black Duck application.
For example, the following curl commands shows how to authenticate with the Black Duck application and get a bearer token prior to making the REST API calls:

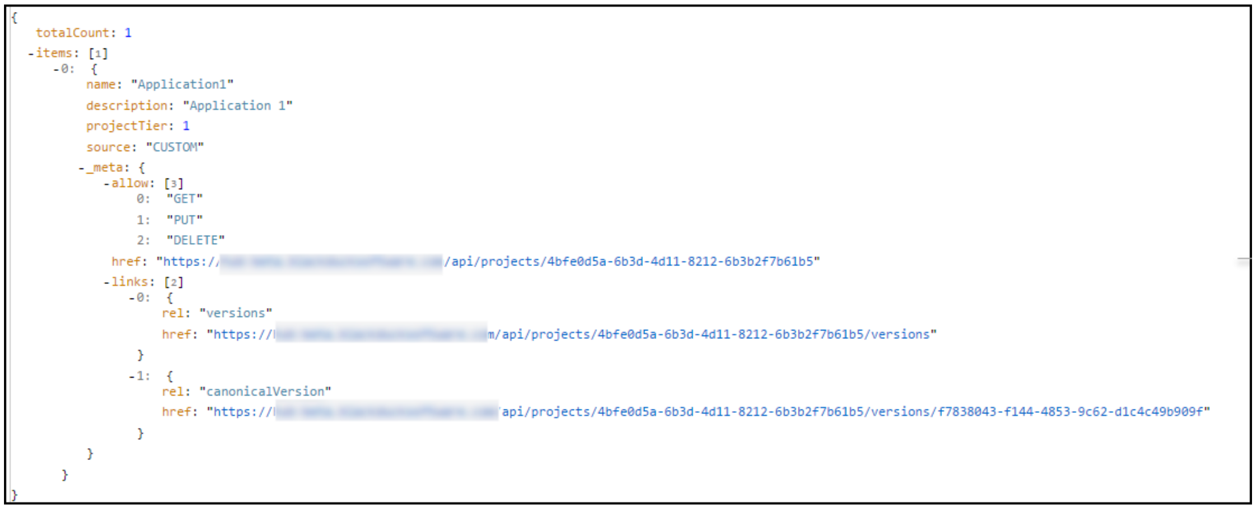
```
curl -X POST \
https://Black Duck/api/tokens/authenticate \
-H 'Accept: application/vnd.blackducksoftware.user-4+json' \
-H 'Authorization: token <API token from Black Duck>' \
```

Using the bearer token that is output when you authenticate, Get a list of projects:

```
curl -X GET \
https://<Black Duck server url>/api/projects \
-H 'Accept: */*' \
-H 'Authorization: Bearer <bearer token>
```

Step 2: Get the Application1 project

```
curl -X GET \
https://<Black Duck server url>:443/api/projects?q=name%3AApplication1
-H 'Accept: */*' \
-H 'Authorization: Bearer <bearer token>
```



```
{
  totalCount: 1
 -items: [1]
   -0:  {
        name: "Application1"
        description: "Application 1"
        projectTier: 1
        source: "CUSTOM"
       -_meta: {
          -allow: [3]
               0:  "GET"
               1:  "PUT"
               2:  "DELETE"
           href: "https://[redacted]/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5"
          -links: [2]
             -0:  {
                  rel: "versions"
                  href: "https://[redacted]m/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions"
               }
             -1:  {
                  rel: "canonicalVersion"
                  href: "https://[redacted]api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions/f7838043-f144-4853-9c62-d1c4c49b909f"
               }
          }
     }
}
```

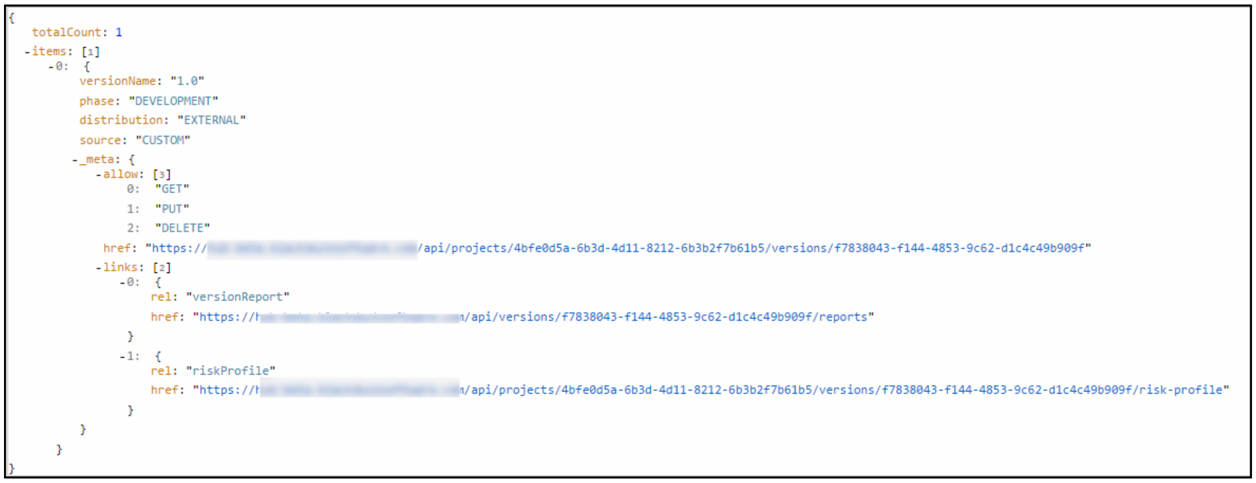
Included in the JSON response are all your available http calls. Notice you can call “versions” or “canonicalVersion”.

Step 3: Get the Application1 project versions

Get the versions of Application1 project.

Using the links provided in the JSON response in step 1 I can use the following to get my project versions:

https://<Black Duck server url>/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions



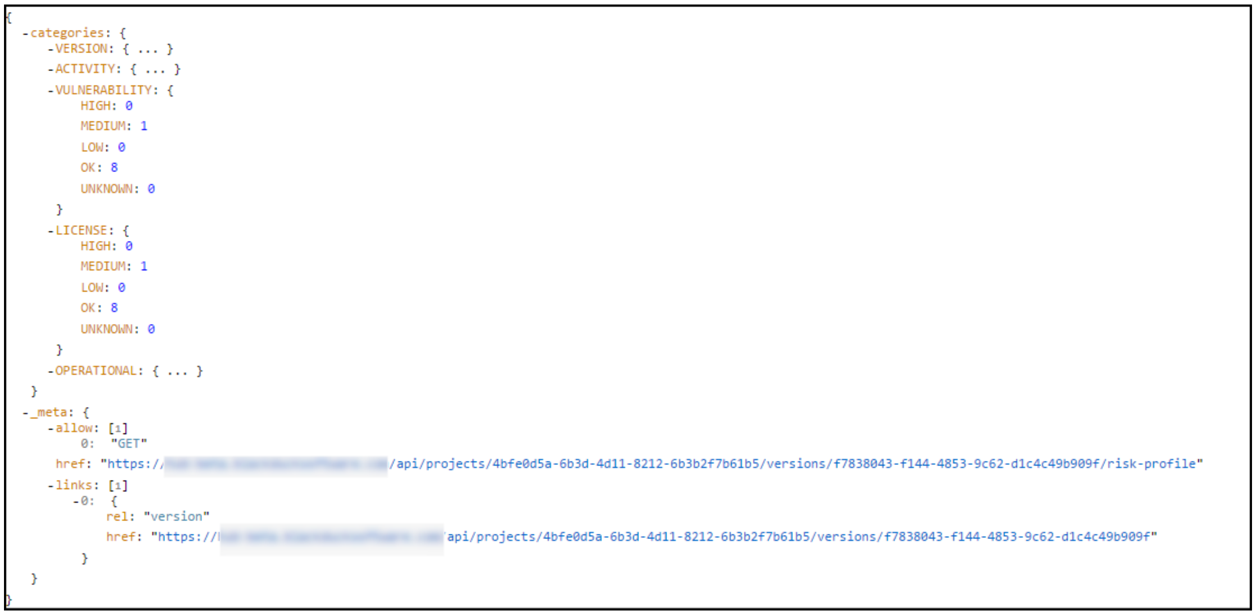
```
{
  totalCount: 1
 -items: [1]
   -0:  {
        versionName: "1.0"
        phase: "DEVELOPMENT"
        distribution: "EXTERNAL"
        source: "CUSTOM"
       -_meta: {
          -allow: [3]
               0:  "GET"
               1:  "PUT"
               2:  "DELETE"
           href: "https://[redacted]/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions/f7838043-f144-4853-9c62-d1c4c49b909f"
          -links: [2]
             -0:  {
                  rel: "versionReport"
                  href: "https://[redacted]n/api/versions/f7838043-f144-4853-9c62-d1c4c49b909f/reports"
               }
             -1:  {
                  rel: "riskProfile"
                  href: "https://[redacted]n/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions/f7838043-f144-4853-9c62-d1c4c49b909f/risk-profile"
               }
          }
     }
}
```

Included in the JSON response are all your available http calls. Notice you can call “riskProfile”.

Step 4: Get the Risk Profile for “1.0” version

Using the links in the JSON response from above I can call the following to get the risk profile:

https://<Black Duck server url>/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions/f7838043-f144-4853-9c62-d1c4c49b909f/risk-profile



```
{
  -categories: {
    -VERSION: { ... }
    -ACTIVITY: { ... }
    -VULNERABILITY: {
        HIGH: 0
        MEDIUM: 1
        LOW: 0
        OK: 8
        UNKNOWN: 0
     }
    -LICENSE: {
        HIGH: 0
        MEDIUM: 1
        LOW: 0
        OK: 8
        UNKNOWN: 0
     }
    -OPERATIONAL: { ... }
  }
  -_meta: {
    -allow: [1]
        0:  "GET"
     href: "https:/[redacted]/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions/f7838043-f144-4853-9c62-d1c4c49b909f/risk-profile"
    -links: [1]
       -0:  {
           rel: "version"
           href: "https://[redacted]api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions/f7838043-f144-4853-9c62-d1c4c49b909f"
        }
  }
}
```

The JSON response includes all risk counts for Version, Activity, Vulnerability, License, and Operational risk. The user can now use those values for whatever external activity they'd like to do (fail a build, create bug tracking tickets, and so on)

Chapter 3: CSRF security

Cross-site request forgeries (CSRF) are types of attacks which perform unwanted actions on a web application without the user's intent. This type of vulnerability happens on web applications which use cookies to identify the user. For example, the attacker uses the same cookies to make a fraudulent request, and attempts to lure the user to click a fraudulent link. If the user clicks the link after being authenticated, the fraudulent request performs any actions available to the logged-in user, but without the knowledge of the user.

The forged request works because browser requests automatically include all credentials associated with the web application, such as the logged-in user's session information in cookies. The server cannot distinguish between the forged request from the user and the legitimate request form the user.

Other names

CSRF attacks are also known by other names; the main difference is the technique used to perform the attack. The names include XSRF, Sea Surf, Session Riding, and Hostile Liking.

CSRF security in Black Duck

Black Duck now features improved security for attempted cross-site request forgeries (CSRF).

Note: CSRF is only needed if the request uses cookies in the request.

Black Duck uses a Synchronized Token Pattern (STP) to address CSRF. This is a technique where a secret and unique token value is passed along with every request through form data or the header. This token is then verified on the server side. If the tokens do not match, then the request is ignored, and a *403 forbidden* error displays. The unique token is generated for each session, not for each request. The same unique token is used for the duration of the session, and a new token is generated for each new session. The token is destroyed when its session is destroyed. The following is an example of the CSRF format:

```
headerName = X-CSRF-TOKEN

parameterName = _csrf

token = 484dcea0-82a7-4f3e-9158-f80513ed86f9
```

Note: The expected authentication method for SDK requests uses API tokens, which do not require CSRF tokens.

When making a REST call to authenticate the user, a POST request (login request) is made to the URL `/j_spring_security_check`. The CSRF information is returned in the response header. The following is an example of the *login request header*.

```
POST /j_spring_security_check HTTP/1.1

Host: qa-hub21.dc1.lan

Connection: keep-alive

Content-Length: 40

Origin: https:

//qa-hub21.dc1.lan

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36

Content-Type: application/x-www-form-urlencoded; charset=UTF-8

Accept: */*

X-Requested-With: XMLHttpRequest

X-FirePHP-Version: 0.0.6

Referer: https://qa-hub21.dc1.lan/

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.8,ja;q=0.6,cs;q=0.4
```

The following is an example of the returned *login response header*. The CSRF token is highlighted.

```
HTTP/1.1 204

Server: nginx/1.10.3

Date: Wed, 14 Jun 2017 16:27:50 GMT

Connection: keep-alive

X-CSRF-HEADER: X-CSRF-TOKEN

X-CSRF-PARAM: _csrf

X-CSRF-TOKEN: 484dcea0-82a7-4f3e-9158-f80513ed86f9

Set-Cookie: AUTHORIZATION_BEARER=<token>

X-Content-Type-Options: nosniff

X-XSS-Protection: 1; mode=block

Cache-Control: no-cache, no-store, max-age=0, must-revalidate

Pragma: no-cache

Expires: 0

X-Frame-Options: SAMEORIGIN
```

To add the CSRF security header to the request:

1. *Only if cookies are used,* copy the CSRF token from the login response header, and include the CSRF token in each subsequent `POST`, `PUT`, `PATCH` and `DELETE` request headers.

The following is a sample request with the CSRF token passed in using the request header:

```
POST /api/v1/projects HTTP/1.1

Host: qa-hub21

Connection: keep-alive

Content-Length: 180

Origin: https://qa-hub21

X-CSRF-TOKEN: 484dcea0-82a7-4f3e-9158-f80513ed86f9

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36

Content-Type: application/json

Accept: application/json, text/javascript, */*; q=0.01

X-Requested-With: XMLHttpRequest

X-FirePHP-Version: 0.0.6

Referer: https://qa-hub21/ui/projects/view:create/action:modal

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.8,ja;q=0.6,cs;q=0.4

Cookie: AUTHORIZATION_BEARER=<token>
```

Note: You must use the CSRF token for POST, PUT, or DELETE requests when using cookies. Any integration with a Black Duck 4.0.0 (or higher) server must include the token.

Chapter 4: API list

The supported REST APIs are documented in the the Black Duck KB user interface. The following lists describe the API endpoints for Black Duck 2019.10.3.

Access the Black Duck API documentation at https://<hub-server>/api-doc/public.html

Make a GET request to the root API endpoint for links to common starting-point API representations.

```
GET /api/
```

Pagination

Requests that return multiple items are paginated by 10 items by default.

Project Endpoints

Creating a project

```
POST /api/projects
```

Listing projects

```
GET /api/projects
```

Reading a single project

```
GET /api/projects/{projectId}
```

Updating a project

```
PUT /api/projects/{projectId}
```

Deleting a project

```
DELETE /api/projects/{projectId}
```

Project Version Endpoints

Creating a project version

```
POST /api/projects/{projectId}/versions
```

Listing project versions

```
GET /api/projects/{projectId}/versions
```

Reading a single project version

```
GET /api/projects/{projectId}/versions/{projectVersionId}
```

Updating a project version

```
PUT /api/projects/{projectId}/versions/{projectVersionId}
```

Deleting a project version

```
DELETE /api/projects/{projectId}/versions/{projectVersionId}
```

Project Assignment Endpoints

Adding a user to a project

```
POST /api/projects/{projectId}/users
```

Listing a project’s assignable users

```
GET /api/projects/{projectId}/assignable-users
```

Reading a single user in a project

```
GET /api/projects/{projectId}/users/{userId}
```

Listing a project’s assigned users

```
GET /api/projects/{projectId}/users
```

Listing projects assigned to a user

```
GET /api/users/{userId}/projects
```

Removing a user from a project

```
DELETE /api/projects/{projectId}/users/{userId}
```

Adding a user group to a project

```
POST /api/projects/{projectId}/usergroups
```

Listing a project's assignable user groups

```
GET /api/projects/{projectId}/assignable-usergroups
```

Reading a single user group in a project

```
GET /api/projects/{projectId}/usergroups/{userGroupId}
```

Listing a project's assigned user groups

```
GET /api/projects/{projectId}/usergroups
```

Listing projects assigned to a user group

GET /api/usergroups/{userGroupId}/projects

Removing a user group from a project

```
DELETE /api/projects/{projectId}/usergroups/{userGroupId}
```

Project Custom Field Endpoints

Reading a single project custom field

```
GET /api/projects/{projectId}/custom-fields/{customFieldId}
```

Updating a project custom field

```
PUT /api/projects/{projectId}/custom-fields/{customFieldId}
```

Updating multiple project custom fields

```
PUT /api/projects/{projectId}/custom-fields
```

Listing project version custom fields

```
GET /api/projects/{projectId}/versions/{projectVersionId}/custom-fields
```

Reading a single project version custom field

```
GET /api/projects/{projectId}/versions/{projectVersionId}/custom-fields/
{customFieldId}
```

Updating a project version custom field

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/custom-fields/
{customFieldId}
```

Updating multiple project version custom fields

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/custom-fields
```

Bill of Materials Endpoints

Adding a component to the BOM

```
POST /api/projects/{projectId}/versions/{projectVersionId}/components
```

Listing BOM components

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components
```

Reading a single BOM component

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}
```

Updating a BOM component

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}
```

Updating a BOM component’s license text

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/licenses/{licenseId}/text
```

Updating a BOM component version’s license text

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/licenses/{licenseId}/text
```

Restoring a BOM component’s license text

```
DELETE /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/licenses/{licenseId}/text
```

Note: Sending a DELETE request to this endpoint removes any custom text and restores this component’s license to its original state.

Restoring a BOM component version’s license text

```
DELETE /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/licenses/{licenseId}/text
```

Note: Sending a DELETE request to this endpoint removes any custom text and restores this component’s license to its original state.

Reading a BOM’s status

```
GET /api/projects/{projectId}/versions/{projectVersionId}/bom-status
```

BOM Comment Endpoints

Adding a comment to a BOM component

```
POST /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/comments
```

Listing BOM component comments

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/comments
```

Reading a single BOM component comment

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/comments/{commentId}
```

Updating a BOM component comment

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/comments/{commentId}
```

Deleting a BOM component comment

```
DELETE /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/comments/{commentId}
```

Adding a comment to a BOM component version

```
POST /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/component-versions/{componentVersionId}/comments
```

Listing BOM component version comments

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/component-versions/{componentVersionId}/comments
```

Reading a single BOM component version comment

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/component-versions/{componentVersionId}/comments/{commentId}
```

Updating a BOM component version comment

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/component-versions/{componentVersionId}/comments/{commentId}
```

Deleting a BOM component version comment

```
DELETE /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/component-versions/{componentVersionId}/comments/{commentId}
```

BOM Custom Field Endpoints

Listing BOM component custom fields

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/custom-fields
```

Reading a single BOM component custom field

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/custom-fields/{customFieldId}
```

Updating a BOM component custom field

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/custom-fields/{customFieldId}
```

Listing Bom component version custom fields

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/custom-fields
```

Reading a single BOM component version custom field

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/custom-fields/{customFieldId}
```

Updating a BOM component version custom field

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/custom-fields/{customFieldId}
```

Bom Policy Endpoints

Reading the policy status of a BOM

```
GET /api/projects/{projectId}/versions/{projectVersionId}/policy-status
```

Reading the policy status of a BOM component

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/policy-status
```

Updating the policy status of a BOM component

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/policy-status
```

Reading the policy status of a BOM component version

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
```

```
{componentId}/versions/{componentVersionId}/policy-status
```

Updating the policy status of a BOM component version

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/policy-status
```

Listing policy rules of a BOM component

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/policy-rules
```

Updating a policy rule of a BOM component

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/policy-rules/{policyRuleId}/policy-status
```

Listing policy rules of a BOM component version

```
GET /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/policy-rules
```

Updating a policy rule of a BOM component version

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/components/
{componentId}/versions/{componentVersionId}/policy-rules/
{policyRuleId}/policy-status
```

Updating the policy status of a file

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/scans/
{scanId}/files/{fileId}/policy-status
```

Updating a policy rule of a file

```
PUT /api/projects/{projectId}/versions/{projectVersionId}/scans/
{scanId}/files/{fileId}/policy-rules/{policyRuleId}/policy-status
```

Component Endpoints

Creating a custom component

```
POST /api/components
```

Searching components

```
GET /api/components
```

Note: The internal version of this does not search, but instead lists custom/modified components.

Reading a single component

```
GET /api/components/{componentId}
```

Updating a component

```
PUT /api/components/{componentId}
```

Deleting a custom component

```
DELETE /api/components/{componentId}
```

Deletes a custom component entirely, or resets modifications on a KnowledgeBase component.

Component Custom Field Endpoints

Reading a single component custom field

```
GET /api/components/{componentId}/custom-fields/{customFieldId}
```

Updating a component custom field

```
PUT /api/components/{componentId}/custom-fields/{customFieldId}
```

Updating multiple component custom fields

```
PUT /api/components/{componentId}/custom-fields
```

Component Version Endpoints

Listing component versions

```
GET /api/components/{componentId}/versions
```

Listing custom component versions

```
GET /api/approval-status/component-versions
```

Creating a custom component version

```
POST /api/components/{componentId}/versions
```

Deleting a custom component version

```
DELETE /api/components/{componentId}/versions/{componentVersionId}
```

Reading a single component version

```
GET /api/components/{componentId}/versions/{componentVersionId}
```

Updating a custom component version

```
PUT /api/components/{componentId}/versions/{componentVersionId}
```

Listing a component version’s cryptographic algorithms

```
GET /api/components/{componentId}/versions/{componentVersionId}/crypto-
algorithms
```

Reading a single component version license

```
GET /api/components/{componentId}/versions/{componentVersionId}/licenses/
{licenseId}
```

Reading a component version’s risk profile

```
GET /api/components/{componentId}/versions/{componentVersionId}/risk-profile
```

Reading a component version license’s text

```
GET /api/components/{componentId}/versions/{componentVersionId}/licenses/
{licenseId}/text
```

Listing remediation options

```
GET /api/components/{componentId}/versions/{componentVersionId}/remediating
```

Component Version Custom Field Endpoints

Reading a Single Component Version Custom Field

```
GET /api/components/{componentId}/versions/{componentVersionId}/custom-fields/
{customFieldId}
```

Updating a Component Version Custom Field

```
PUT /api/components/{componentId}/versions/{componentVersionId}/custom-fields/
{customFieldId}
```

Updating Multiple Component Version Custom Fields

```
PUT /api/components/{componentId}/versions/{componentVersionId}/custom-fields
```

License Endpoints

Creating a license

```
POST /api/licenses
```

Listing licenses

```
GET /api/licenses
```

Reading a single license

`GET /api/licenses/{licenseId}`

Updating a license

`PUT /api/licenses/{licenseId}`

Deleting a license

`DELETE /api/licenses/{licenseId}`

Reading a license’s text

`GET /api/licenses/{licenseId}/text`

Updating a license’s text

`PUT /api/licenses/{licenseId}/text`

License Families

Listing license families

`GET /api/license-families`

Reading a single license family

`GET /api/license-families/{licenseFamilyId}`

Creating a license family

`POST /api/license-families`

Updating a license family

`PUT /api/license-families/{licenseFamilyId}`

Deleting a license family

`DELETE /api/license-families/{licenseFamilyId}`

License Terms

Adding a license term to a license

`POST /api/licenses/{licenseId}/license-terms`

Listing license terms of a license

```
GET /api/licenses/{licenseId}/license-terms
```

Reading a single license term of a license

```
GET /api/licenses/{licenseId}/license-terms/{licenseTermId}
```

Updating a license term of a license

```
PUT /api/licenses/{licenseId}/license-terms/{licenseTermId}
```

Deleting a license term from a license

```
DELETE /api/licenses/{licenseId}/license-terms/{licenseTermId}
```

Vulnerability Endpoints

Listing vulnerabilities by component

```
GET /api/components/{componentId}/vulnerabilities
```

Listing vulnerabilities by component version

```
GET /api/components/{componentId}/versions/
{componentVersionId}/vulnerabilities
```

Listing vulnerabilities by component version and origin

```
GET /api/components/{componentId}/versions/{componentVersionId}/origin/
{componentVersionOriginId}/vulnerabilities
```

Reading a single vulnerability

```
GET /api/vulnerabilities/{vulnerabilityId}
```

Common weakness enumeration endpoints

Endpoints involving common weakness enumerations (CWE).

Reading a single CWE

```
GET /api/cwes/{cweId}
```

Affected Project Version Endpoints

Listing affected project versions

```
GET /api/vulnerabilities/{vulnerabilityId}/affected-projects
```

Notifications

Notifications are records of application events of interest to end users.

Listing notifications

```
GET /api/notifications
```

Reading a single notification

```
GET /api/notifications/{notificationId}
```

Policies

Creating a policy

```
POST /api/policy-rules
```

Listing policies

```
GET /api/policy-rules
```

Reading a single policy

```
GET /api/policy-rules/{policyRuleId}
```

Updating a policy

```
PUT /api/policy-rules/{policyRuleId}
```

Deleting a policy

```
DELETE /api/policy-rules/{policyRuleId}
```

Reports

Version Report Endpoints

Listing version reports

```
GET /api/versions/{projectVersionId}/reports
```

Reading a single version report

```
GET /api/versions/{projectVersionId}/reports/{reportId}
```

Creating a version report

```
POST /api/versions/{projectVersionId}/reports
```

Deleting a version report

```
DELETE /api/versions/{projectVersionId}/reports/{reportId}
```

Version License Report Endpoints

Listing version license reports

```
GET /api/versions/{projectVersionId}/license-reports
```

Reading a single version license report

```
GET /api/versions/{projectVersionId}/license-reports/{reportId}
```

Creating a version license report

```
POST /api/versions/{projectVersionId}/license-reports
```

Deleting a version license report

```
DELETE /api/versions/{projectVersionId}/license-reports/{reportId}
```

Vulnerability Report Endpoints

Reading a single vulnerability report

```
GET /api/vulnerability-reports/{reportId}
```

Deleting a vulnerability report

```
DELETE /api/vulnerability-reports/{reportId}
```

Listing vulnerability remediation reports

```
GET /api/vulnerability-remediation-reports
```

Creating a vulnerability remediation report

```
POST /api/vulnerability-remediation-reports
```

Listing vulnerability status reports

```
GET /api/vulnerability-status-reports
```

Creating a vulnerability status report

```
POST /api/vulnerability-status-reports
```

Listing vulnerability update reports

```
GET /api/vulnerability-update-reports
```

Creating a vulnerability update report

```
POST /api/vulnerability-update-reports
```

Reading a report’s contents

```
GET /api/reports/{reportId}/contents
```

Scans

Uploading analysis files

```
POST /api/scan/data/
```

Note: Synopsys recommends that you do not upload files manually using this API; use tools such as Detect to both generate and upload files.

Code Location Endpoints

Listing code locations

```
GET /api/codelocations
```

Reading a single code location

```
GET /api/codelocations/{codeLocationId}
```

Updating a code location

```
PUT /api/codelocations/{codeLocationId}
```

Deleting a code location

```
DELETE /api/codelocations/{codeLocationId}
```

Listing code location scans

```
GET /api/codelocations/{codeLocationId}/scan-summaries
```

Reading a single scan

```
GET /api/scan-summaries/{scanId}
```

Status Related Endpoints

Status related endpoints provide information about the application such as linking, job information, and registration.

Registration Endpoints

Activating registration

```
POST /api/registration
```

Updating registration

```
PUT /api/registration
```

Job Endpoints

Listing jobs

```
GET /api/jobs
```

Reading a single job

```
GET /api/jobs/{jobId}
```

Rescheduling a job

```
PUT /api/jobs/{jobId}
```

Terminating a job

```
DELETE /api/jobs/{jobId}
```

User Endpoints

Reading the current user

```
GET /api/current-user
```

Creating a user

```
POST /api/users
```

Listing users

```
GET /api/users
```

Reading a single user

```
GET /api/users/{userId}
```

Updating a user

```
PUT /api/users/{userId}
```

Resetting a user’s password

```
PUT /api/users/{userId}/resetpassword
```

Changing a user’s password

```
PUT /api/users/{userId}/changepassword
```

User Group Endpoints

Creating a user group

```
POST /api/usergroups
```

Listing user groups

```
GET /api/usergroups
```

Reading a single user group

```
GET /api/usergroups/{userGroupId}
```

Updating a user group

```
PUT /api/usergroups/{userGroupId}
```

Deleting a user group

```
DELETE /api/usergroups/{userGroupId}
```

Role Endpoints

Listing roles

```
GET /api/roles
```

Reading a single role

```
GET /api/roles/{roleId}
```

Listing a Role's Inheriting Users

```
GET /api/roles/{roleId}/inheriting-users
```

Listing a role's users

```
GET /api/roles/{roleId}/users
```

Listing a user's roles

```
GET /api/users/{userId}/roles
```

Listing a user's inherited roles

```
GET /api/users/{userId}/inherited-roles
```

Assigning roles to a user

```
POST /api/users/{userId}/roles
```

Reading a user's assigned role

```
GET /api/users/{userId}/roles/{roleAssignmentId}
```

Deleting a user's assigned role

```
DELETE /api/users/{userId}/roles/{roleAssignmentId}
```

Administration Endpoints

Listing custom field datatypes

```
GET /api/custom-fields/types
```

Listing custom field objects

```
GET /api/custom-fields/objects
```

Reading a single custom field object

```
GET /api/custom-fields/objects/{customFieldObject}
```

Creating a custom field

```
POST /api/custom-fields/objects/{customFieldObject}/fields
```

Listing custom fields

```
GET /api/custom-fields/objects/{customFieldObject}/fields
```

Reading a single custom field

```
GET /api/custom-fields/objects/{customFieldObject}/fields/{customFieldId}
```

Updating a custom field

```
PUT /api/custom-fields/objects/{customFieldObject}/fields/{customFieldId}
```

Deleting a custom field

```
DELETE /api/custom-fields/objects/{customFieldObject}/fields/{customFieldId}
```

Note: By deleting a custom field definition you also delete any of that custom field's options.

Listing custom field options

```
GET /api/custom-fields/objects/{customFieldObject}/fields/
{customFieldId}/options
```

Creating a custom field option

```
POST /api/custom-fields/objects/{customFieldObject}/fields/
{customFieldId}/options
```

Updating a custom field option

```
PUT /api/custom-fields/objects/{customFieldObject}/fields/
{customFieldId}/options/{customFieldOptionId}
```

Deleting a custom field option

```
DELETE /api/custom-fields/objects/{customFieldObject}/fields/
{customFieldId}/options/{customFieldOptionId}
```