Getting Started with the SDK

Black Duck SCA 2025.1.1

06-03-2025

Contents

Preface

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 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.
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.

The installation methods for installing Black Duck software in a Kubernetes or OpenShift environment are Helm. Click the following links to view the documentation.

- Helm is a package manager for Kubernetes that you can use to install Black Duck. Black Duck supports Helm3 and the minimum version of Kubernetes is 1.13.

Black Duck integration documentation is available on:

- https://sig-product-docs.blackduck.com/bundle/detect/page/integrations/integrations.html
- https://documentation.blackduck.com/category/cicd_integrations

Customer support

If you have any problems with the software or the documentation, please contact Black Duck Customer Support:

- Online: https://community.blackduck.com/s/contactsupport
- To open a support case, please log in to the Black Duck Community site at https://community.blackduck.com/s/contactsupport.
- Another convenient resource available at all times is the online Community portal.

Black Duck Community

The Black Duck 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 Black Duck 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 Black Duck 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 Black Duck 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 Black Duck experts and our Knowledgebase.
- Share – Collaborate and connect with Black Duck 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@blackduck.com.

Training

Black Duck Customer Education 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.

In Black Duck Education, 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://blackduck.skilljar.com/page/black-duck or for help with Black Duck, select **Black Duck Tutorials** from the Help menu (?) in the Black Duck UI.

Black Duck Statement on Inclusivity and Diversity

Black Duck is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

Black Duck Security Commitments

As an organization dedicated to protecting and securing our customers' applications, Black Duck is equally committed to our customers' data security and privacy. This statement is meant to provide Black Duck customers and prospects with the latest information about our systems, compliance certifications, processes, and other security-related activities.

This statement is available at: Security Commitments | Black Duck

1. Accessing 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 do one of the following:

- Access the Black Duck API documentation at https://<hub-server>/api-doc/public.html
- From the help menu (Help menu) located on the top navigation bar in the Black Duck UI, select **REST API Developers Guide**.

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

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

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.bdio+zip

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.system-announcement-1+json

application/vnd.blackducksoftware.user-4+json application/vnd.blackducksoftware.vulnerability-4+json

multipart/form-data

text/plain

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::

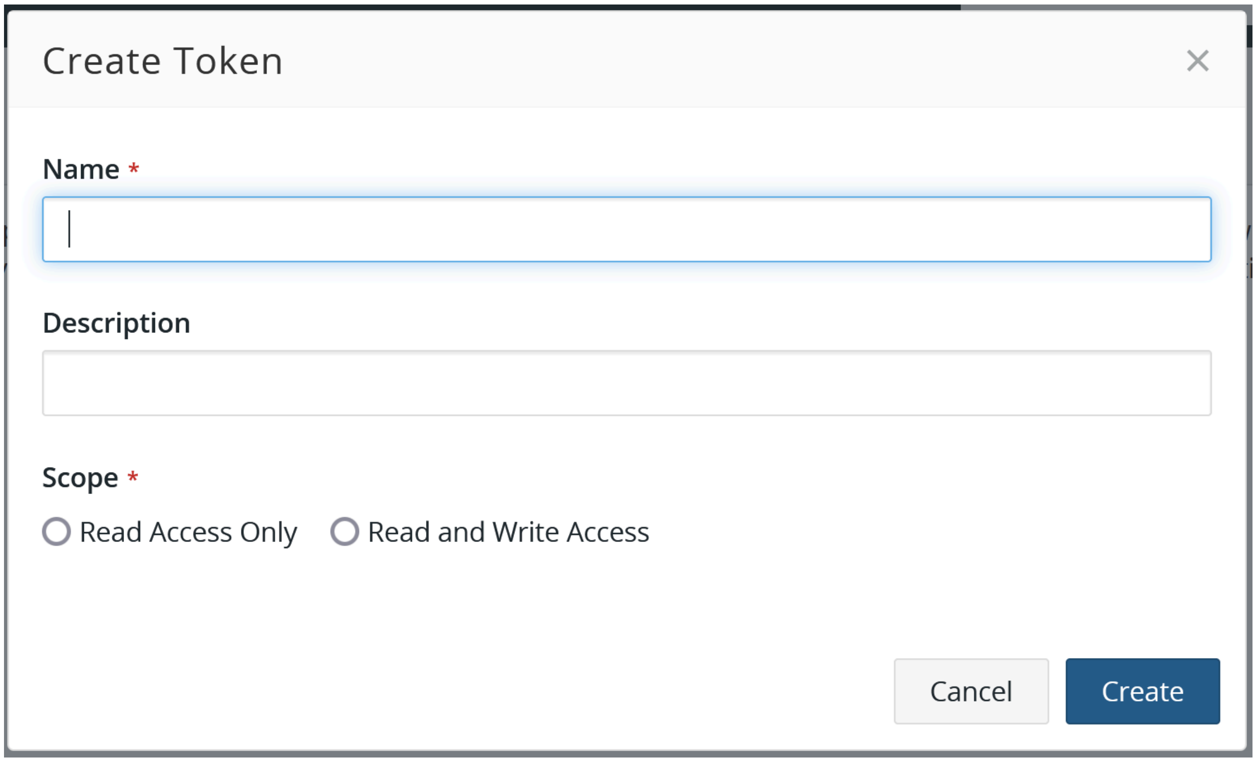
1. Generate an API token in Black Duck by going to the Black Duck UI, and from the user menu located on the top navigation bar, select **My Access Tokens** to open the My Access Tokens page where you generate your API token.
2. Pass the API token in an HTTP POST to `/api/tokens/authenticate` to generate 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. From the user menu located on the top navigation bar, select **My Access Tokens**. The My Access Tokens page appears.
2. Click **Create New Token**. The Create New Token dialog box appears.

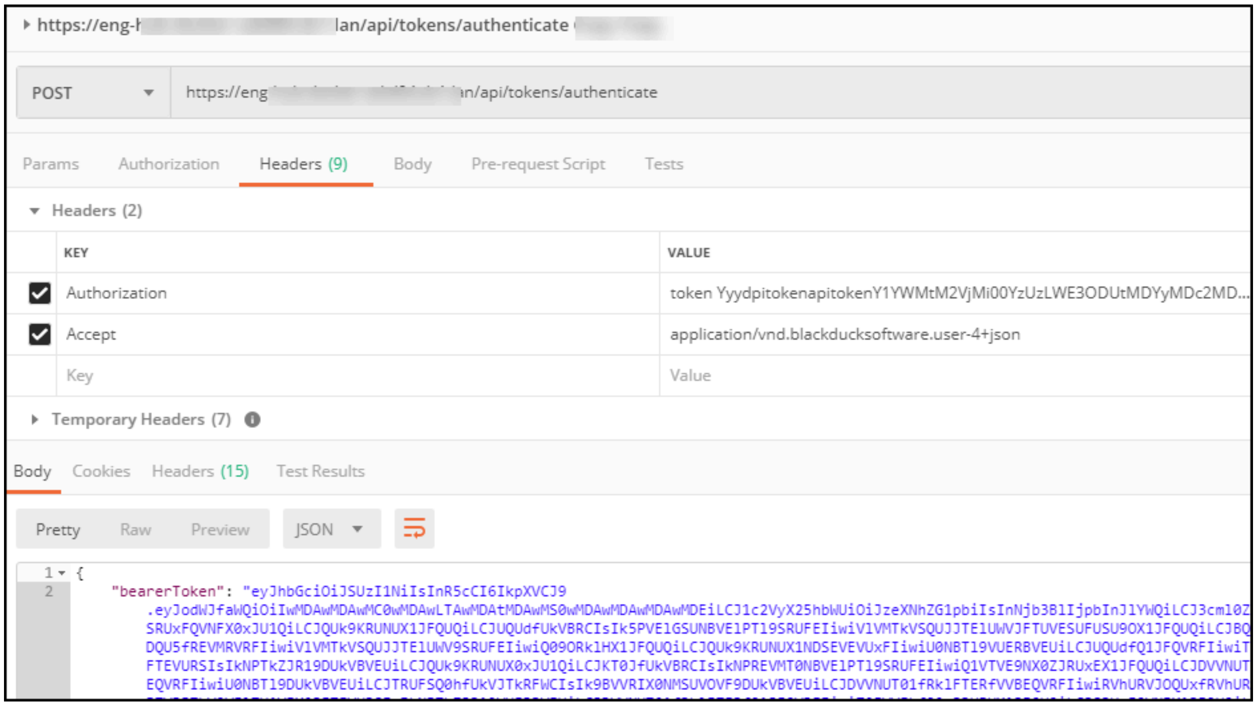


3. Type a name in the **Name** field.
4. **Optional:** In the **Description** field, type a description or definition.
5. Select **Read Access** and/or **Write Access**.
6. Click **Create**. The API token displays in a pop-up window. For security reasons, this is the only time your user API token displays. Please save this token. If the token is lost, you must regenerate it.
7. **Optional:** To modify an access token that you created, click the arrow in the same row as the access token name to open a drop-down menu and select **Edit**, **Delete**, or **Regenerate**.

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

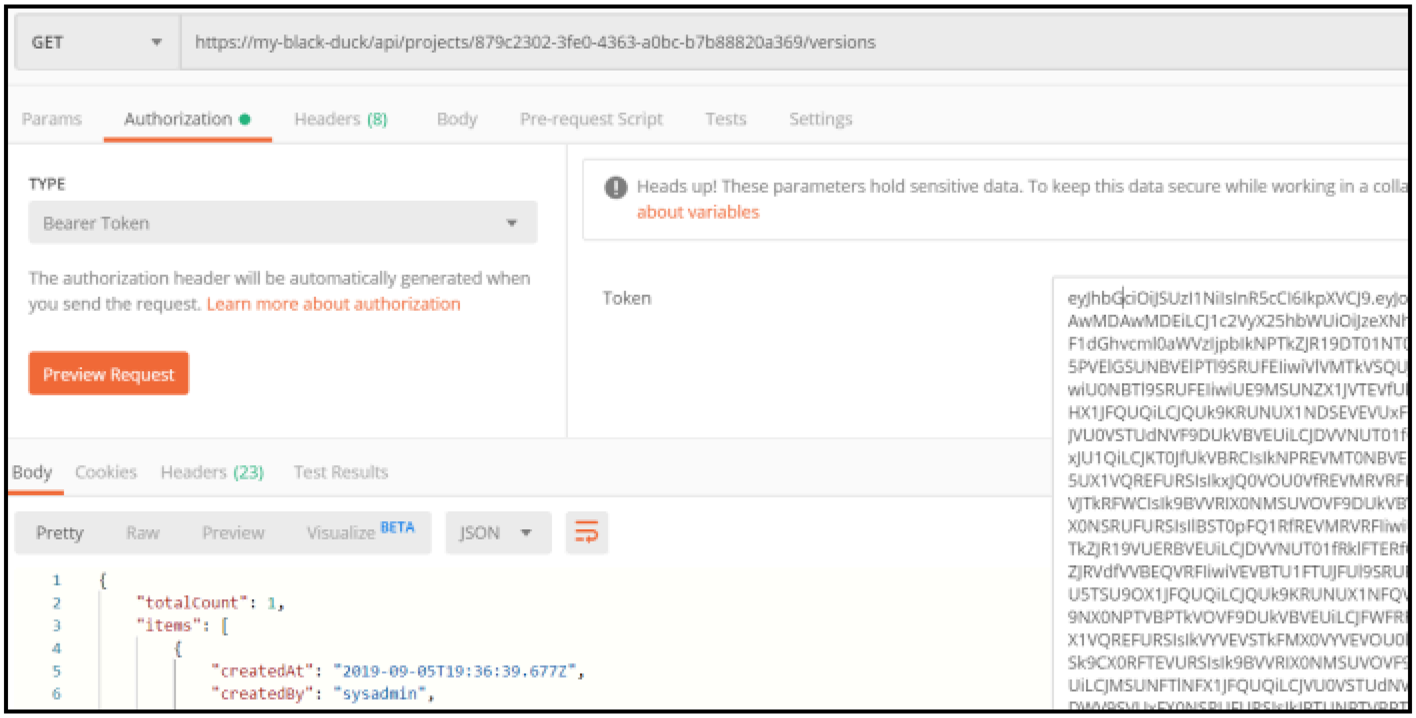
```
{
    "bearerToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJodWJfaWQiOiIwMDAwMDAwMC0wMDAwLTAwMDAtMDAwMS0wMDAwMDAwMDAwMDEiLCJ1c2VyX25hbWUiOiJzeXNhZG1pbiIsInNjb3BlIjpbInJ1YWQiLCJ3cml0Z
        SRUxFQVNFX0xJU1QiLCJQUk9KRUNUX1JFQUQiLCJUQUdfUkVBRCIsIk5PVE1GSUNBVE1PT19SRUFEIiwiV1VMTkVSQUJJTE1UWVJFTUVESUFUSU9OX1JFQUQiLCJBQ
        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 \
https://<Black Duck server URL>/api/tokens/authenticate \
-H "Accept: application/vnd.blackducksoftware.user-4+json" \
-H "Authorization: token
 YjU4ODkyNmItODI1Ny00NjRkLWJlNDEtMWE0MzE3MmFiODgwOmMxMzNkMzkNTdjOQ=="
```

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
5PVEIGSUNBVEIPTI9SRUFEIiwiVlVMTkVSQU
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

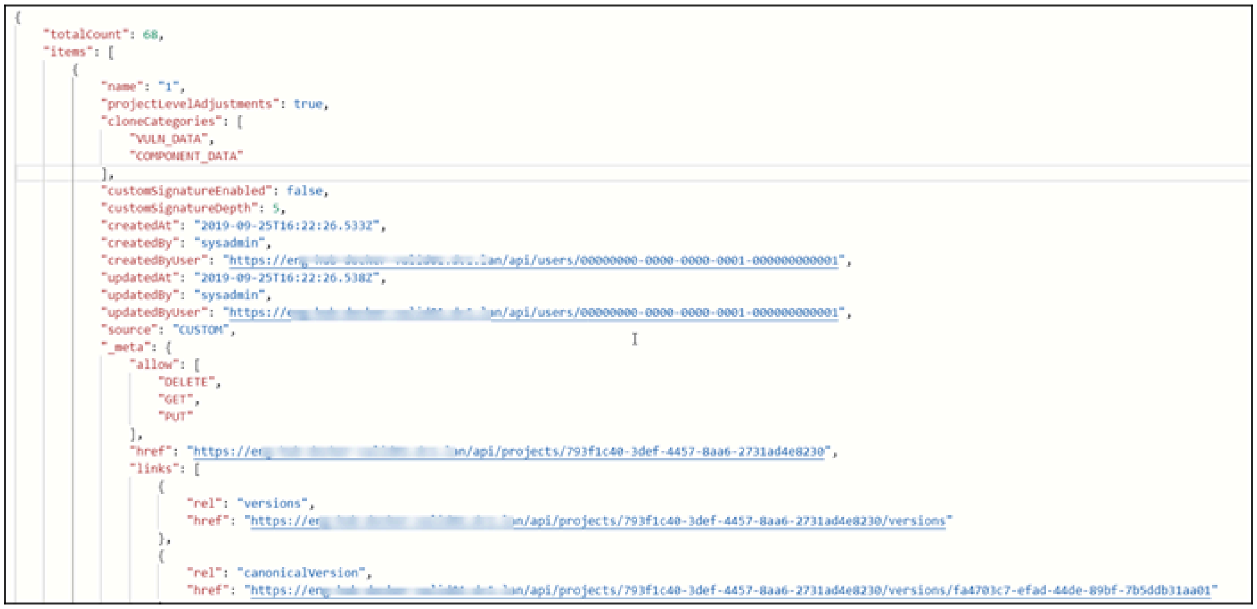
Additionally, no CSRF, (as described here) is included since it is not required when using the ‘Authorization’ header to pass the token.

```
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:

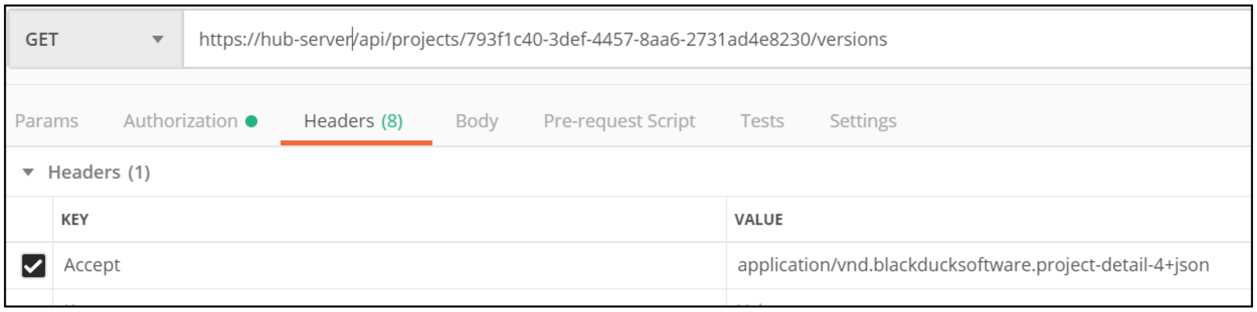
```
curl -X GET \
https://<Black Duck server URL>/api/projects/ \
-H "Accept: application/vnd.blackducksoftware.project-detail-4+json" \
-H "Authorization: Bearer
 eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJodWJfaWQiOiIwMDAwMDAwMC0wMDAwLTAwMDAtMDAwMS0wMDAwMDAwMDAwM
```

The following image shows 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://en[redacted]an/api/projects/793f1c40-3def-4457-8aa6-2731ad4e8230",
                "links": [
                    {
                        "rel": "versions",
                        "href": "https://en[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

	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.

Changing the expiration time for a bearer token

To extend the expiration time of a bearer token used in REST API, use the `docker-compose.local-overrides.yml` file to override the default setting by configuring the `HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIRE` environment variable with the new expiration value in seconds.

The `HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIRE` property is the number of seconds that the access tokens take to expire.

Note: The expiration configuration change only works for API tokens that are created after you change the setting in the `docker-compose.local-overrides.yml` file. The expiration time that you configure isn't updated for existing database records/API tokens when the setting is changed and the service is restarted.

Creating a Postman Collection

Create a Postman Collection in Black Duck for import into the Postman application.

1. Log into Black Duck.

2. Open a browser tab and paste the following URL using you Black Duck server address.

   `https://<your_black_duck_server>/api-doc/postman-collection-public.json`

3. On the page that's generated, right-click and save as `postman-collection-public.json`
4. Import the saved `postman-collection-public.json` into your Postman application.

Generating OpenAPI endpoints

For users using OpenAPI Specification (OAS), you can generate customer-facing endpoints through `/api-doc/openapi3-public.json`.

1. Log into Black Duck.
2. Open a browser tab and paste the following URL using you Black Duck server address.

   `https://<your_black_duck_server>/api-doc/openapi3-public.json`

3. On the page that's generated, right-click and save as `openapi3-public.json`
4. Import the saved `openapi3-public.json` into your application.

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:

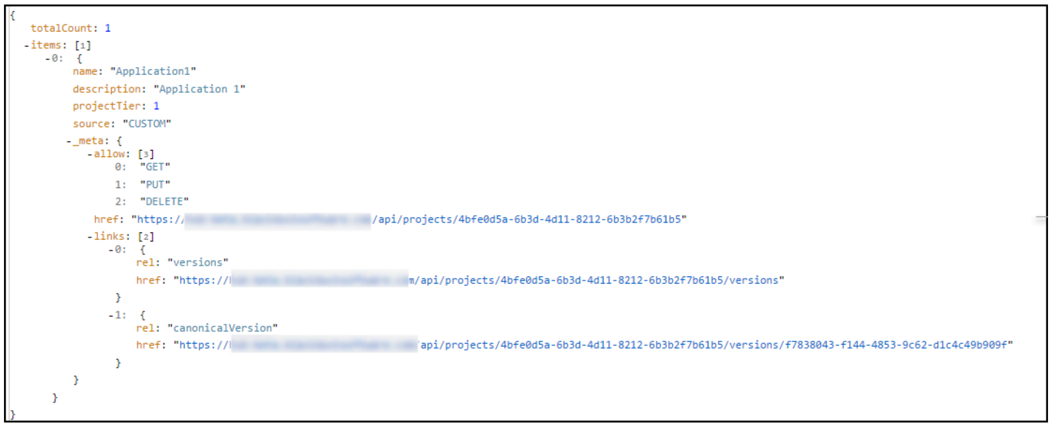
```
curl -X POST \
https://<Black Duck server URL>/api/tokens/authenticate \
-H "Accept: application/vnd.blackducksoftware.user-4+json" \
-H "Authorization: token <API token from Black Duck>"
```

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>/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"
              }
        }
    }
}
```

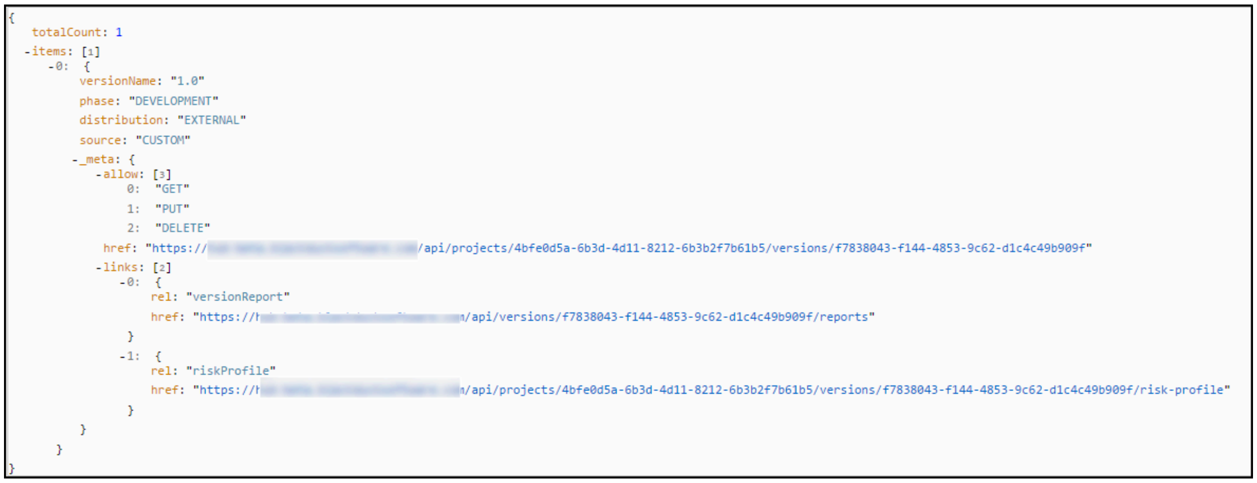
Included in the JSON response are all your available http calls. Notice you can call “versions” or “canonicalVersion”.

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



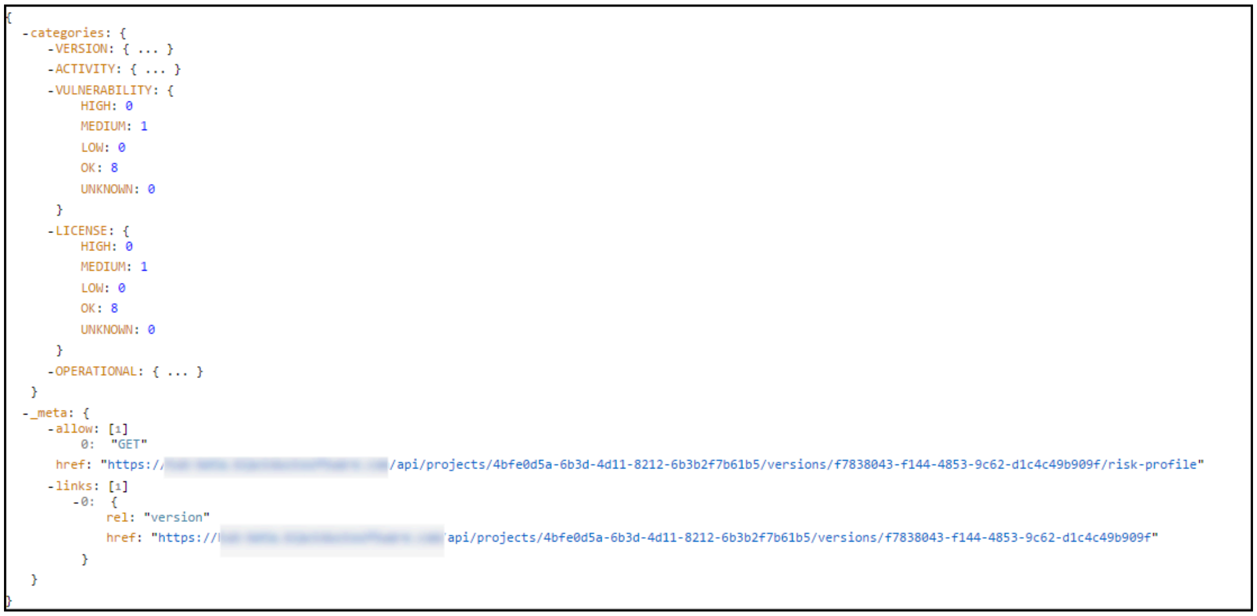
```
{
  totalCount: 1
 -items: [1]
   -0:  {
        versionName: "1.0"
        phase: "DEVELOPMENT"
        distribution: "EXTERNAL"
        source: "CUSTOM"
      -_meta: {
         -allow: [3]
             0:  "GET"
             1:  "PUT"
             2:  "DELETE"
          href: "https://[redacted]/api/projects/4bfe0d5a-6b3d-4d11-8212-6b3b2f7b61b5/versions/f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

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.

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.

4. API list

For more information on API requests, please refer to the REST API Developers Guide available in Black Duck.