CxSAST v9.2.0

API Guide

Contents

CxSAST API Guide

This API guide includes information about developing client implementations using the CxSAST API.

Before creating CxSAST API clients, it is recommended to become familiar with the CxSAST concept.

CxSAST (OData) API

The following section includes information about developing client implementations using the CxSAST OData API. Before creating API clients, it is recommended to become familiar with CxSAST concepts.

CxSAST (OData) API Authentication

Authentication and Permissions

An authenticated user can be granted access to restricted sets of data and benefit from extended quotas for API calls. Starting from this version (v9.0.0), the CxSAST (Odata) API features an authentication mechanism for users to be granted their specific authorizations.

To complete the authentication procedure, you need to know the basics of the following:

- The basics of using Odata APIs, e.g. requests, responses, etc.
- The basics of using Checkmarx Access Control and CxSAST

We will use access token-based authentication in Postman for this example.

Permissions

To perform the authentication procedure and implement Odata API functionality you also require the following permission:

- Use Odata

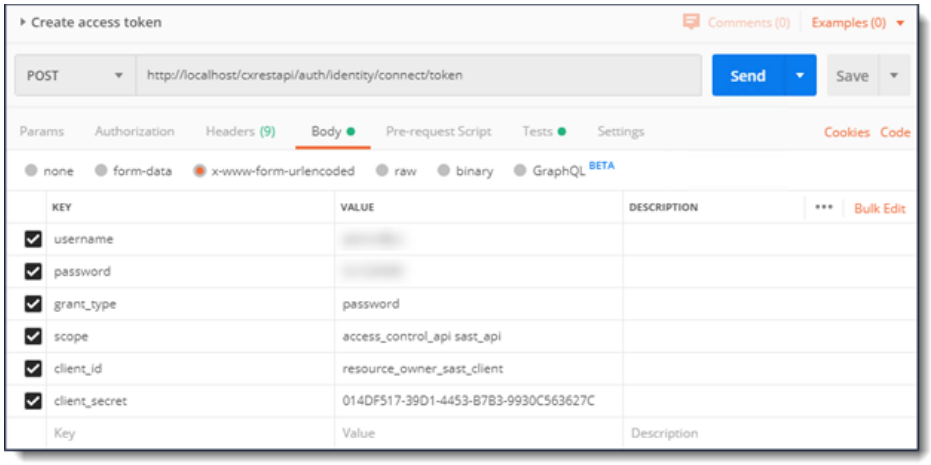
Step 1: Requesting an access token for authentication

In order to complete the authentication procedure, you first need to receive an access token for authentication. The first thing you need to do is make a request to the authentication server by including the credentials received from the authorization server.

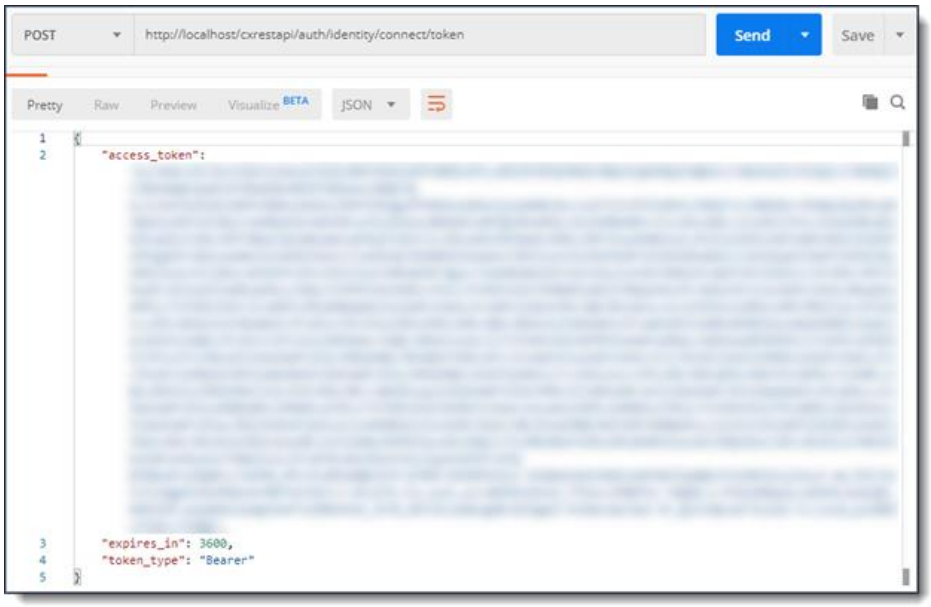
To do this, just submit (POST) the desired credentials to the token endpoint using the following application/x-www-form-urlencoded format in the request body:

- Endpoint example: http://<server-name/ip>:<port>/cxrestapi/auth/identity/connect/token.
- Credentials example:
- username: <Cx username>
- password: <Cx password>
- grant_type: Value must be set as 'password'
- scope: Value must be set as 'access_control_api sast_api
- client_id: Value must be set as 'resource_owner_sast_client'
- client_secret: Default value is set as '014DF517-39D1-4453-B7B3-9930C563627C'

The access token request contains the following Postman format:



The POST request described above creates a login session and returns the requested access token response information, which will look similar to the following Postman example:



If the access token request is valid and authorized, the authorization server issues an access token response.

A successful response appears in the following format:

```
{
      "access_token": "eyJ0eXAiOiJKV1QiLCJhbGtr0hoV4Vj8GNkyk2A……..",
      "expires in": 3600,
      "token_type": "Bearer"
}
```

If the request failed client authentication or is invalid, the authentication server returns an access token error response.

An error response appears in the following format:

```
{
      "error":"invalid_grant"
}
```

This error could mean that the provided authorization grant is invalid, expired, revoked or issued to another client.

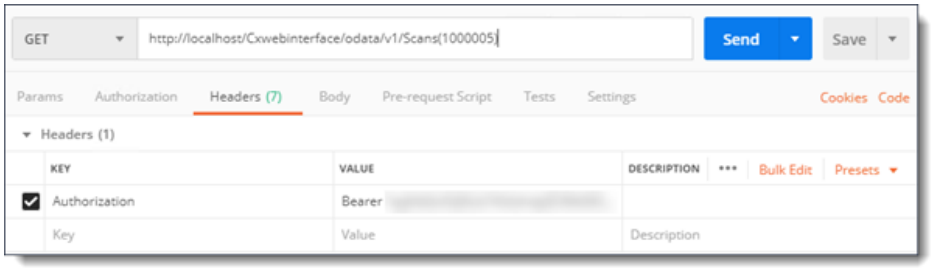
Step 2: Using the access token in a request to the resource server

To make a request to the resource server, send the access token (access_token) received during authentication.

To do this, submit (GET) the access token value to the resource in the request header:

- Resource example: http://<rver-name/ip>:<port>/Cxwebinterface/odata/v1/Scans(<scan_Id>)
- Credentials example:
    - Authorization: Bearer <access token value>

The session request contains the following Postman format:



When you submit the request, this access token value is used for authentication until such a time that the token expires.

Token Expiration

A token automatically expires, if not used for a predefined amount of time (i.e. expires in = 3600 secs). You will receive an error response letting you know that your token is invalid. In this case, you will need to re-authenticate using the access token request for authentication procedure in Step 1.

CxSAST (OData) API Overview & Examples

Overview

The CxSAST OData API enables retrieving data from the CxSAST database.

Data Consumed

The data that can currently be consumed from the CxSAST Database includes:

- Projects
- Scans
- Results

Metadata

The $metadata is an OData service. It returns an EDMX (Entity Data Model XML) document that contains a complete description of the feeds, types, properties, and relationships exposed by the service in EDM (Entity Data Model).

For details about the actual data structure exposed by CxSAST, see http://localhost/Cxwebinterface/odata/v1/$metadata.

Authentication

The required authentication method depends on the method used by Cx Server - either Basic or Windows authentication (up to v8.9.0).

For v9.0.0 and up, token-based authentication using the new access control module is used. For more information about authentication using access control, refer to CxSAST (OData) API Authentication (v9.0.0 and up).

OData.Org

For further information on how to implement the functions of OData, see the OData.org website.

Examples

For a specific scan Id:

- Request result: retrieve all data for a specific scan Id:
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Scans(1000005)

Retrieve number of LOC scanned for a specific scan:

- Request result: retrieve LOC scanned value for a specific scan Id
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Scans(1000005)?$select=LOC

Retrieve number of LOC scanned for all scan:

- Request result: retrieve LOC scanned value for all scans
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Scans?$select=LOC,Id

Top 5 Projects By Risk Score:

- Requested result: list the 5 Projects whose most recent scans yielded the highest Risk Score
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan&$orderby=LastScan/RiskScore%20desc&$top=5

Top 5 Projects By (Last) Scan Duration:

- Requested result: list the 5 Projects whose most recent scan had the longest duration
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan&$orderby=LastScan/ScanDuration%20desc&$top=5

All Projects with their Last Scan and the High Vulnerabilities:

- Requested result: list all projects, and for each project list the security issues (vulnerabilities) with a High severity degree found in the project's most recent scan.
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan($expand=Results($filter=Severity%20eq%20CxDataRepository.Severity%27High%27))
- Same query broken into two pages of 10 projects each:

1. http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan($expand=Results($filter=Severity%20eq%20CxDataRepository.Severity%27High%27))&$top=10&skip=0
2. http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan($expand=Results($filter=Severity%20eq%20CxDataRepository.Severity%27High%27))&$top=10&skip=10

For a specific Project, for the LastScan, get two pages of 10 results each:

1. Obtain the ProjectId either by looping through projects or via apriori knowledge (using CxWebClient or other means): http://localhost.checkmarx.net/Cxwebinterface/odata/v1/Projects?$select=Id
2. Assuming that ProjectId = 4205 from previous step, get the Scanid of LastScan: http://localhost.checkmarx.net/Cxwebinterface/odata/v1/Projects(4205)/LastScan?$select=Id
3. Assuming that ScanId = 1006992 from previous step, get the Results in two pages of 10 results each:

    a) http://localhost.checkmarx.net/Cxwebinterface/odata/v1/Scans(1006992)/Results?$top=10&$skip=0
    b) http://localhost.checkmarx.net/Cxwebinterface/odata/v1/Scans(1006992)/Results?$top=10&$skip=10

Only projects that have high vulnerabilities (in the Last Scan):

- Requested result:list only projects that had vulnerabilities with a High severity degree found in their last scan
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan($expand=Results)&$filter=LastScan/Results/any(r:%20r/Severity%20eq%20CxDataRepository.Severity%27High%27)

For a specific project, retrieve all scans within a predefined time range and their H/M/L values:

- Requested result:list all scans carried out in a specific project within a predefined time range, as well as their H/M/L (High/Medium/Low) values. The sample query below refers to a time range between the 23/07/2015 and 23/08/2015.
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects(11)/Scans?$filter=ScanRequestedOn%20gt%202015-07-23%20and%20ScanRequestedOn%20lt%202015-08-23&$select=Id,ScanRequestedOn,High,Medium,Low&$orderby=ScanRequestedOn%20desc

For all projects in a team, return the number of recurrent/resolved/new issues (vulnerabilities) within a predefined time range:

- Requested result:list the number of recurrent/resolved/new issues (vulnerabilities) detected by scans made in all projects that were carried out in a team within a predefined time range. The sample query below refers to a time range between the 23/07/2015 and 23/08/2015.
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects?$filter=OwningTeamId%20eq%20%2700000000-1111-1111-b111-989c9070eb11%27&$expand=Scans($expand=ResultSummary;$select=Id,ScanRequestedOn,ResultSummary;$filter=ScanRequestedOn%20gt%202015-07-23%20and%20ScanRequestedOn%20lt%202015-08-23)

For a specific project, the state of each scan result for a specific project since a specific date:

- Requested result: for a specific project, list all the scans starting from a specific date, and for each scan retrieve three parameters (Id, ScanId, and StateId) as well as the state of each of the scan's vulnerabilities that was found in scans since the specified date.
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Scans?$filter=ProjectId%20eq%2011%20and%20ScanRequestedOn%20gt%202014-12-31&$expand=Results($expand=State;$select=Id,ScanId,StateId)

Retrieve a count (not in JSON format) of the projects in the system:

- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects/$count

Retrieve all projects with a custom field that has a specific value:

- Requested result: retrieve all projects that contain a custom filed (for example, *ProjManager*, indicating the project manager's name) with a specific value (for example, Joe).
- Query used for retrieving the data:http://localhost/Cxwebinterface/odata/v1/Projects?$filter=CustomFields/any(f: f/FieldName eq 'ProjManager' and f/FieldValue eq 'Joe')

Retrieve all projects with a custom field, as well as the custom field's information:

- Requested result: retrieve all projects that contain a custom field (for example, *ProjManager*, indicating the project manager's name), as well as the custom field's information.
- Query used for retrieving the data: http://localhost/cxwebinterface/odata/v1/Projects?$expand=CustomFields&$filter=CustomFields/any(f: f/FieldName eq 'Field1')

Retrieve the query (SQL Injection, etc.) that was run for a particular unique scan result:

- Requested result: selects a particular unique scan result and lists the query (SQL Injection, etc.) that was run
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Results(Id=18,ScanId=1000001)?$expand=Query($select=Name)

Retrieve a list of presets associated with each project:

- Requested result: retrieves a list of presets associated with each project
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects?$expand=Preset

Retrieve all projects that are set up with a non-standard configuration:

- Requested result: retrieve all projects that are set up with a non-standard configuration, such as “Multi-Lanaguage Scan (v8.4.2 and up)".
- Query used for retrieving the data: http://localhost/Cxwebinterface/odata/v1/Projects?$filter=EngineConfigurationId or http://localhost/Cxwebinterface/odata/v1/Projects?$filter=EngineConfigurationId%20gt%201

Possible Issues

The following issues may occur when working with OData:

- The default value of the timeout parameter can be too low when executing highly complex queries. For details, see the Troubleshooting section OData Commands Time Out.
- The process may run out of memory after consuming the .NET 4GB of memory. For details, see the Troubleshooting section .NET Memory Limit Exceeded.

Extending Timeout

The timeout parameter defines how long the code waits for SQL to execute the query.

While simple queries can be executed successfully and free the database within the limits of a low timeout value, such a low value can cause highly complex queries to fail, resulting in an http 400 status (Bad Request) message. This message notifies that the query execution has timed out.

When such a message is received, users should first ensure that the queries are fine-tuned and do not unnecessarily load the system. If all fine-tuning efforts do not yield the requested timeout decrease, users can enable handling complex queries by modifying the OData command timeout parameter. This parameter is set through the key specified below, whose default value is 120 seconds.

CxComponentConfiguration Table

The record with [Key] = 'ODATA_COMMAND_TIMEOUT'

.NET Memory Limit Exceeded

When a command does not time out but the **.NET 4GB of memory runs out**, the user receives the following response with an **http 400 status** (Bad Request):

"The result is too large to return, the process ran out of memory. Try requesting paged data or refining your search".

Solution:

When retrieving large data quantities, it is recommended to use OData in paging mode. This significantly reduces the initial response time, thereby preventing overload on the CX server machine.

http://www.odata.org/documentation/odata-version-2-0/uri-conventions/

Request broken into two pages of 10 projects each:

- http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan($expand=Results($filter=Severity%20eq%20CxDataRepository.Severity%27High%27))&$top=10&skip=0
- http://localhost/Cxwebinterface/odata/v1/Projects?$expand=LastScan($expand=Results($filter=Severity%20eq%20CxDataRepository.Severity%27High%27))&$top=10&skip=10

CxSAST (REST) API

Checkmarx CxSAST is a unique source code analysis solution that provides tools for identifying, tracking, and repairing technical and logical flaws in the source code, such as security vulnerabilities, compliance issues, and business logic problems. Using the CxSAST Auditor tool, you can configure your own additional queries for security, QA, and business logic purposes.

CxSAST provides scan results either as static reports, or in an interactive interface that enables tracking runtime behavior per vulnerability through the code, and provides tools and guidelines for remediation. Results can be customized to eliminate false positives, and various types of workflow metadata can be added to each result instance. Metadata is maintained through subsequent scans, as long as the instance continues to be found.

CxSAST includes a REST-based CxSAST API that supports the latest version of the REST protocol. The CxSAST (REST) API provides the ability to manage all CxSAST related tasks. The following data can be consumed through the CxSAST (REST) API; Login, Projects, Scans, Scan Results, Scan Reports, Engines, Managing Users, Data Retention and Open Source Analysis. For more information about the CxSAST (REST) API, refer to the CxSAST (REST) API Summary.

CxSAST (REST) API Authentication

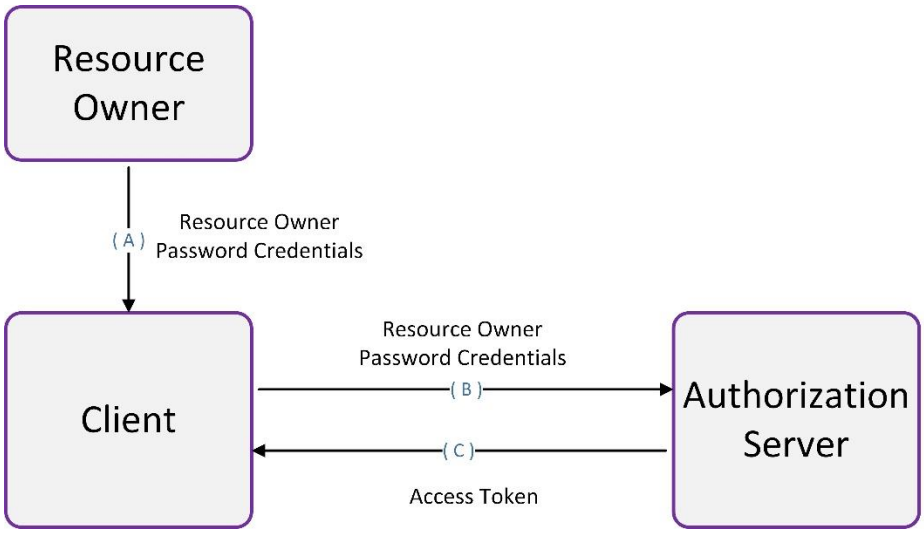
This section includes the CxREST API Authentication options.

OAuth 2.0 Authentication Concept

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

Resource Owner Password Credentials Grant

Currently CxSAST supports the OAuth 2.0 resource owner password credentials grant type. This grant type is used in cases where the resource owner has a trust relationship with the client. The authorization server takes special care when enabling this grant type and only allows it when other flows are not viable.



The above diagram illustrates the following flow steps:

- The resource owner provides the client with its username and password.
- The client requests an access token from the authentication server's token endpoint by including the credentials received from the resource owner. When making the request, the client authenticates with the authentication server.
- The authorization server authenticates the client and validates the resource owner credentials, and if valid, issues an access token.

The client discards the credentials (username and password) once an access token has been obtained.

Prerequisites

- Checkmarx CxSAST (v8.6.0 or higher) installed
- OAuth2 and SSO - If SSO is already configured on the system the Internet Information Services Manager (IIS) might need additional configuration as explained:

    a) Go to **Start > Search > IIS** and open the Internet Information Services Manager.
    b) From **Sites > Default Web Site > CxRestAPI**, select **Authentication** and enable **Anonymous Authentication**.

- OAuth2 validity is set at 24 hours (default).

    c) Confirm that these are the settings, if not, update them accordingly.
    d) Restart the IIS service in order for the changes to take effect.

- Configuration Keys – The following default value is provided in the CxComponentConfiguration database table in order to use the CxSAST (REST) API:

  IdentityAuthority – In case of using HTTP (instead of HTTPs) or the machine name is not used as the Checkmarx domain (for example, when using a load balancer), the IdentityAuthority must be changed.

Authentication Procedure

For more information about the authentication procedure when using tokens, refer to Using the CxSAST (REST) API (v8.6.0 and up).

CxSAST (REST) API Summary

This section provides a basic summary of our CxSAST (REST) API offering. CxSAST (REST) APIs are grouped according to their product area and each API has a direct link to the relevant API documentation. Cx version, indicating when the API was first introduced, as well as the API version is also indicated.

Group	API	Cx Ver.	API Ver.
Login	POST /auth/login	8.6.0 up	0.1/1.0/1.1/2.0
Projects	GET /projects	8.6.0 up	1.0/2.0
	POST /projects	8.6.0 up	1.0/2.0
	GET /projects/{id}	8.6.0 up	1.0/2.0
	PUT /projects/{id}	8.8.0 up	1.0/2.0
	PATCH /projects/{id}	8.7.0 up	1.0/2.0
	DELETE /projects/{id}	8.7.0 up	1.0/2.0
	POST /projects/{id}/branch	8.8.0 up	1.0
	GET /issueTrackingSystems	8.7.0 up	1.0
	GET /issueTrackingSystems/{id}/metadata	8.7.0 up	1.0
	GET /projects/{id}/sourceCode/excludeSettings	8.7.0 up	1.0
	PUT /projects/{id}/sourceCode/excludeSettings	8.7.0 up	1.0

Group	API	Cx Ver.	API Ver.
	GET /projects/{id}/sourceCode/remoteSettings/git	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/git	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/svn	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/svn	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/tfs	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/tfs	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/custom	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/custom	8.7.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/shared	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/shared	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/perforce	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/perforce	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/git/ssh	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/svn/ssh	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/attachments	8.6.0 up	1.0
	GET /customTasks	8.7.0 up	1.0
	GET /customTasks/{id}	8.7.0 up	1.0
	GET /customFields	8.8.0 up	1.0
	POST /projects/{id}/dataRetentionSettings	8.7.0 up	1.0
	POST /projects/{id}/issueTrackingSettings/jira	8.7.0 up	1.0
	GET /sast/presets	8.6.0 up	1.0
	GET /sast/presets/{id}	8.6.0 up	1.0

Group	API	Cx Ver.	API Ver.
Scans	GET /sast/scans	8.8.0 up	1.0
	POST /sast/scans	8.6.0 up	1.0
	GET /sast/scans/{id}	8.6.0 up	1.0
	PATCH /sast/scans/{id}	8.7.0 up	1.0
	DELETE /sast/scans/{id}	8.8.0 up	1.0
	GET /sast/scans/{id}/resultsStatistics	8.8.0 up	1.0
	GET /sast/scansQueue/{id}	8.7.0 up	1.0
	PATCH /sast/scansQueue/{id}	8.7.0 up	1.0
	GET /sast/scansQueue	8.5.0 up	0.1/1.0
	GET /sast/scanSettings/{projectId}	8.6.0 up	1.0/1.1
	POST /sast/scanSettings	8.6.0 up	1.0/1.1
	PUT /sast/scanSettings	8.7.0 up	1.1
	PUT /sast/project/{projectId}/scheduling	8.8.0 up	1.0
Scan Results	POST /sast/results/tickets	8.8.0 up	1.0
	POST /sast/projects/{id}/publisher/policyFindings	8.9.0 up	1.0
	GET /sast/projects/{id}/publisher/policyFindings/status	8.9.0 up	1.0
Scan Reports	GET /reports/sastScan/{id}	8.6.0 up	1.0
	GET /reports/sastScan/{id}/status	8.6.0 up	1.0
	POST /reports/sastScan	8.6.0 up	1.0
Engines	GET /sast/engineServers	8.5.0 up	0.1/1.0
	POST /sast/engineServers	8.5.0 up	0.1/1.0
	DELETE /sast/engineServers/{id}	8.5.0 up	0.1/1.0

Group	API	Cx Ver.	API Ver.
	GET /sast/engineServers/{id}	8.5.0 up	0.1/1.0
	PUT /sast/engineServers/{id}	8.5.0 up	0.1/1.0
	GET /sast/engineConfigurations	8.6.0 up	1.0
	GET /sast/engineConfigurations/{id}	8.6.0 up	1.0
Queries	GET /Queries/{queryid}/CxDescription	8.6.0 up	1.0
Managing Users	GET /auth/teams	8.6.0 up	1.0
Data Retention	POST /sast/dataRetention/stop	8.8.0 up	1.0
	POST /sast/dataRetention/byDateRange	8.8.0 up	1.0
	POST /sast/dataRetention/byNumberOfScans	8.8.0 up	1.0
	GET /sast/dataRetention/{requestId}/status	8.9.0 up	1.1
Open Source Analysis	GET /osa/scans	8.4.2 up	1.0/2.0
	GET /osa/scans/{scanId}	8.4.2 up	1.0/2.0
	POST /osa/scans	8.4.2 up	1.0/2.0
	GET /osa/fileextensions	8.6.0 up	1.0
	GET /osa/licenses	8.6.0 up	1.0
	GET /osa/libraries	8.6.0 up	1.0/2.0
	GET /osa/vulnerabilities	8.4.2 up	1.0
	GET /osa/reports	8.4.2 up	1.0

To access a live Swagger environment navigate to: http://<ServerName>/cxrestapi/help/swagger/ui/index (e.g. http://localhost/cxrestapi/help/swagger/ui/index)

CxSAST (REST) API Summary (replacement for v9.0.0 changes)

This section provides a basic summary of our CxSAST (REST) API offering. CxSAST (REST) APIs are grouped according to their product area and each API has a direct link to the relevant API documentation. Cx version, indicating when the API was first introduced, as well as the API version is also indicated.

Group	API	Cx Ver.	API Ver.
Login	POST /auth/login	8.6.0 to 8.9.0	0.1/1.0/1.1/2.0
Projects	GET /projects	8.6.0 up	1.0/2.0
	POST /projects	8.6.0 up	1.0/2.0
	GET /projects/{id}	8.6.0 up	1.0/2.0
	PUT /projects/{id}	8.8.0 up	1.0/2.0
	PATCH /projects/{id}	8.7.0 up	1.0/2.0
	DELETE /projects/{id}	8.7.0 up	1.0/2.0
	POST /projects/{id}/branch	8.8.0 up	1.0
	GET /issueTrackingSystems	8.7.0 up	1.0
	GET /issueTrackingSystems/{id}/metadata	8.7.0 up	1.0
	GET /projects/{id}/sourceCode/excludeSettings	8.7.0 up	1.0
	PUT /projects/{id}/sourceCode/excludeSettings	8.7.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/git	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/git	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/svn	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/svn	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/tfs	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/tfs	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/custom	8.7.0 up	1.0

Group	API	Cx Ver.	API Ver.
	POST /projects/{id}/sourceCode/remoteSettings/custom	8.7.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/shared	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/shared	8.6.0 up	1.0
	GET /projects/{id}/sourceCode/remoteSettings/perforce	8.7.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/perforce	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/git/ssh	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/remoteSettings/svn/ssh	8.6.0 up	1.0
	POST /projects/{id}/sourceCode/attachments	8.6.0 up	1.0
	GET /customTasks	8.7.0 up	1.0
	GET /customTasks/{id}	8.7.0 up	1.0
	GET /customFields	8.8.0 up	1.0
	POST /projects/{id}/dataRetentionSettings	8.7.0 up	1.0
	POST /projects/{id}/issueTrackingSettings/jira	8.7.0 up	1.0
	GET /sast/presets	8.6.0 up	1.0
	GET /sast/presets/{id}	8.6.0 up	1.0
Scans	GET /sast/scans	8.8.0 up	1.0
	POST /sast/scans	8.6.0 up	1.0
	GET /sast/scans/{id}	8.6.0 up	1.0
	PATCH /sast/scans/{id}	8.7.0 up	1.0
	DELETE /sast/scans/{id}	8.8.0 up	1.0
	GET /sast/scans/{id}/resultsStatistics	8.8.0 up	1.0
	GET /sast/scansQueue/{id}	8.7.0 up	1.0

Group	API	Cx Ver.	API Ver.
	PATCH /sast/scansQueue/{id}	8.7.0 up	1.0
	GET /sast/scansQueue	8.5.0 up	0.1/1.0
	GET /sast/scanSettings/{projectId}	8.6.0 up	1.0/1.1
	POST /sast/scanSettings	8.6.0 up	1.0/1.1
	PUT /sast/scanSettings	8.7.0 up	1.0/1.1
	PUT /sast/project/{projectId}/scheduling	8.8.0 up	1.0
Scan Results	POST /sast/results/tickets	8.8.0 up	1.0
	POST /sast/projects/{id}/publisher/policyFindings	8.9.0 up	1.0
	GET /sast/projects/{id}/publisher/policyFindings/status	8.9.0 up	1.0
	GET /sast/scans/{id}/results/{pathId}/shortDescription	9.0.0 up	1.0
Scan Reports	GET /reports/sastScan/{id}	8.6.0 up	1.0
	GET /reports/sastScan/{id}/status	8.6.0 up	1.0
	POST /reports/sastScan	8.6.0 up	1.0
Engines	GET /sast/engineServers	8.5.0 up	0.1/1.0
	POST /sast/engineServers	8.5.0 up	0.1/1.0
	DELETE /sast/engineServers/{id}	8.5.0 up	0.1/1.0
	GET /sast/engineServers/{id}	8.5.0 up	0.1/1.0
	PUT /sast/engineServers/{id}	8.5.0 up	0.1/1.0
	GET /sast/engineConfigurations	8.6.0 up	1.0
	GET /sast/engineConfigurations/{id}	8.6.0 up	1.0
Managing Users	GET /auth/teams	8.6.0 to 8.9.0	1.0
Data Retention	POST /sast/dataRetention/stop	8.8.0 up	1.0/1.1

Group	API	Cx Ver.	API Ver.
	POST /sast/dataRetention/byDateRange	8.8.0 up	1.0/1.1
	POST /sast/dataRetention/byNumberOfScans	8.8.0 up	1.0/1.1
	GET /sast/dataRetention/{requestId}/status	8.9.0 up	1.0/1.1
Open Source Analysis	For more information about APIs for working with Open Source Analysis (CxOSA) tasks, see CxOSA (REST) API Summary.		

To access a live Swagger environment navigate to: http://<ServerName>/cxrestapi/help/swagger/ui/index (e.g. http://localhost/cxrestapi/help/swagger/ui/index)

Using the CxSAST (REST) API

This section explains how to use the CxSAST (REST) API.

General Points

In this part of the documentation, we will use the words “CxSAST service” to refer to the base address of the REST service. For CxSAST, this URL can be found on the **/<server-name/ip>:<port>/cxrestapi** path, relative to the platform hostname (**http://<host name>:<port>/cxrestapi**).

Unless stated otherwise, all addresses in the rest of this documentation are relative to the CxSAST service, apart from when requesting an access token for authentication.

Versioning and Media Type

The CxSAST is installed with the latest version (i.e. v=1.0) of the CxSAST (REST) API. In order to use another version of the CxSAST (REST) API you will need to add '**;v=<version>**' to a media type header in the request. The media type header defined will depend on the request method used:

- GET Request – Accept: application/json;v=1.0
- POST, PUT, PATCH and DELETE Requests – Content-Type: application/json;v=1.0

- Not specifying the version automatically applies the latest default version and may cause your script/code to break.

Origin

In order to be compliant with an audit trail, you will need to add 'cxOrigin=<request_origin>' to the cxOrigin header in the request (e.g. Media Type = cxOrigin: cx-jenkins').

Specifying the origin ensures that all CxSAST (REST) API requests are logged to the audit trail with their original source. Not specifying the origin will automatically apply the default origin (i.e. other).

Format

The CxSAST service supports json format.

Testing

We will use authentication and request submission in Postman for most of the CxSAST (REST) API testing examples.

Authentication

An authenticated user can be granted access to restricted sets of data and benefit from extended quotas for API calls. The CxSAST (REST) API features an authentication mechanism for users to be granted their specific authorizations.

- An Active Directory user can't be used for this API. Even when SSO is enabled for the web client, you must use an application user and not an active directory user for this API.

To complete the authentication procedure you must be familiar with the basics:

- The basics of using REST APIs, e.g. requests, responses, headers
- The basics of using CxSAST

We will use access token-based authentication in Postman for this example.

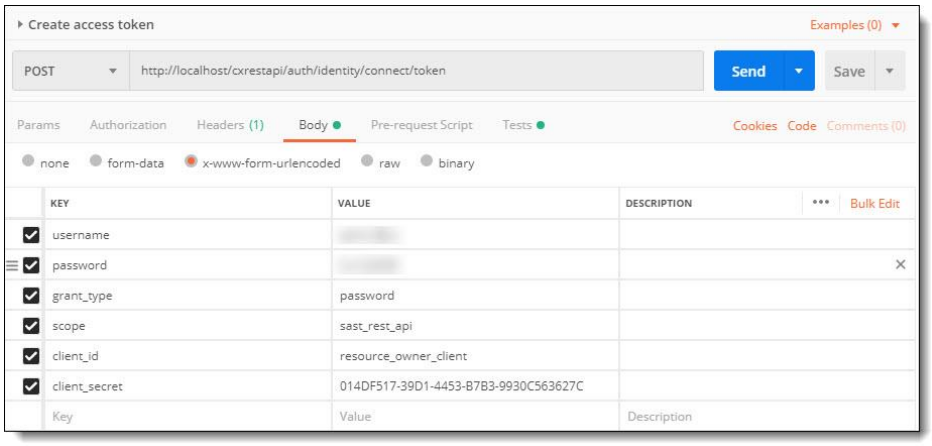
Step 1: Requesting an access token for authentication

You need to receive an access token for authentication. The first thing you need to do is make a request to the authentication server by including the credentials received from the authorization server.

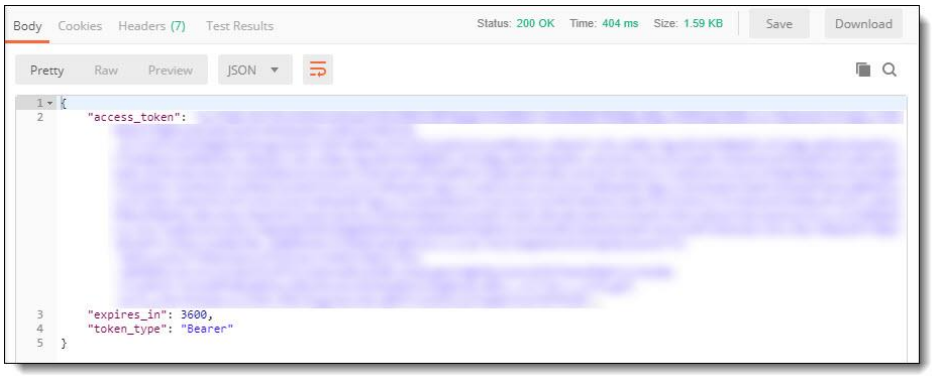
To do this, just submit (POST) the desired credentials to the token endpoint using the application/x-www-form-urlencoded format in the request body:

- Endpoint example: http://<server-name/ip>:<port>/cxrestapi/auth/identity/connect/token
- Credentials example:
    - username: <Cx username>
    - password: <Cx password>
    - grant_type: Value must be set as 'password'
    - scope: Value must be set as 'sast_rest_api'
    - client_id: Value must be set as 'resource_owner_client'
    - client_secret: Value must be set as '014DF517-39D1-4453-B7B3-9930C563627C'

The access token request looks similar to the following:



This creates a login session and returns the requested access token response information, which will look similar to the following:



If the access token request is valid and authorized, the authorization server issues an access token response.

An example for a successful access token response:

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGtr0hoV4Vj8GNkyk2A……..",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

If the request failed client authentication or is invalid, the authentication server returns an access token error response.

An example for an access token error response:

```
{
  "error":"invalid_grant"
}
```

This error means that the provided authorization grant is invalid, expired, revoked or issued to another client. For more examples of errors with responses, see the Error Responses.

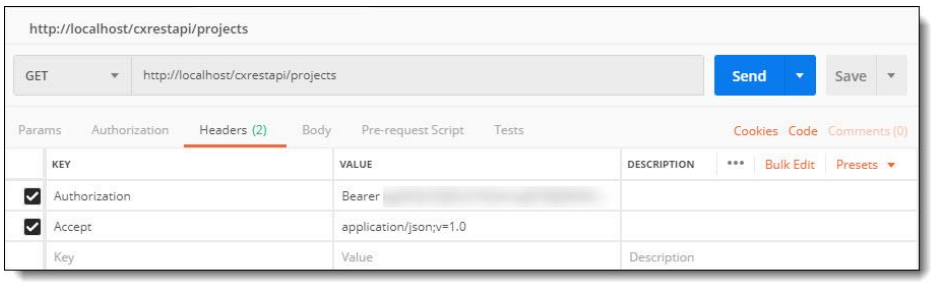
Step 2: Using the access token in a request to the resource server

When you want to make a request to the resource server, you should send the access token (access_token) received during authentication.

To do this, just submit (GET) the access token value to the resource in the request header:

- Resource example: http://<server-name/ip>:<port>/cxrestapi/projects
- Credentials example:
    - Authorization: Bearer <access token value>
    - Accept: application/json;v=1.0

The session request looks similar to the following Postman example:



When you submit the request, this access token value is used for authentication, until the token expires.

Token Expiration

If you have not used the CxSAST (REST) API for a while, you need to perform the authentication again because your token has expired. You will receive an error response. The response body will contain a message telling you that your token is invalid. At this point, you will need to re-authenticate using the access token request procedure in the CxSAST (REST) API.

Authorization Errors

In case of an error the authorization server responds with an HTTP 400 (Bad Request) status code (unless specified otherwise) and includes the following error with responses:

error – A single error code with one of the following:

- invalid_request – The request is missing a required parameter, includes an unsupported parameter, repeats a parameter or includes multiple credentials
- invalid_client – Client authentication failed (e.g., unknown client, no client authentication included, or unsupported authentication method)
- invalid_grant – The provided authorization grant is invalid, expired, revoked or was issued to another client
- unauthorized_client – The authenticated client is not authorized to use this authorization grant type
- unsupported_grant_type – The authorization grant type is not supported by the authorization server
- invalid_scope – The requested scope is invalid, unknown or malformed.

Authentication Log

```
The authentication log can be found in the WebAPIAll file
(<directory>:\Program Files\Checkmarx\Logs\WebAPI\Trace). The
following is an example of the log:

2017-11-02 09:33:02,331 [19] INFO - Start token request
2017-11-02 09:33:02,331 [19] INFO - Secret id found:
resource_owner_client
2017-11-02 09:33:02,347 [19] INFO - Client validation success
2017-11-02 09:33:02,347 [19] INFO - Start token request validation
2017-11-02 09:33:02,347 [19] INFO - Start password token request
validation
```

```
2017-11-02 09:33:02,362 [19] INFO - REST login: Login Ended
successfully. Username:
203186009069174177246150222065055136232142232254
2017-11-02 09:33:02,362 [19] INFO - Password token request validation
success.
2017-11-02 09:33:02,362 [19] INFO - Token request validation success
{
  "ClientId": "resource_owner_client",
  "ClientName": "Cx Resource Owner Client",
  "GrantType": "password",
  "Scopes": "sast_rest_api",
  "UserName": "<username>",
  "AuthenticationContextReferenceClasses": [],
  "Raw": {
    "username": "<username>",
    "password": "******",
    "grant_type": "password",
    "scope": "sast_rest_api",
    "client_id": "resource_owner_client",
    "client_secret": "******"
  }
}
2017-11-02 09:33:02,362 [19] INFO - Creating token response
2017-11-02 09:33:02,362 [19] INFO - Processing token request
2017-11-02 09:33:02,768 [19] INFO - End token request
2017-11-02 09:33:02,768 [19] INFO - Returning token response.
```

CxSAST (REST) API – Projects

This section covers all project related API calls.

Get All Project Details - GET /projects

Get details of all visible projects. The retrieval of project details is performed after creating a project. To create a project use the POST/projects API.

Usage

1. POST/projects and create new project with default configuration settings
2. GET /projects and get details of all projects

URL

http://localhost/cxrestapi/projects

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=2.0

Parameters

Optional:

projectName=[string] – Unique name of a specific project or projects
teamId=[string] – Unique Id of a specific team or teams.

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept:
application/json' --header 'Authorization: Bearer <access token> -d '{
\
   "name": "Project 5", \
   "owningTeam": 1, \
   "isPublic": true \
 }' 'http://localhost/cxrestapi/projects'
```

Query String

Optional (example only):

/projects?projectName=myProject&teamId=00000000-1111-1111-b111-989c9070eb11

Sample Response

```
[
    {
        "id": 1,
        "teamId": 1,
        "name": "Project 1 (CxTechDocs)",
        "isPublic": true,
        "sourceSettingsLink": {
            "type": "local",
            "rel": "source",
            "uri": null
        },
        "link": {
            "rel": "self",
            "uri": "/projects/1"
        }
    },
    {
        "id": 2,
        "teamId": 1,
        "name": "Project 2 (CxTechDocs)",
```

```
        "isPublic": true,
        "sourceSettingsLink": {
            "type": "local",
            "rel": "source",
            "uri": null
        },
        "link": {
            "rel": "self",
            "uri": "/projects/2"
        }
    },
    {
        "id": 3,
        "teamId": 1,
        "name": "Project 4 (CxTechDocs)",
        "isPublic": true,
        "sourceSettingsLink": {
            "type": "local",
            "rel": "source",
            "uri": null
        },


        "link": {
            "rel": "self",
            "uri": "/projects/3"
        }
    }
]
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves details of all visible projects. If values for both parameters (projectName and teamId) are provided, the response is filtered according to the defined parameters. If no parameter values are provided full results are returned in the response. If the request fails, it returns an error response. Project Id (id) can be used to get details of a specific project using GET/projects/{id}. Team Id (id) can be used to create a new project with default configuration setting using POST /projects.

Create Project with Default Configuration - POST /projects

This section covers REST APIs for working with creating projects.

Create Project with Default Configuration - POST /projects

Create a new project with default preset and configuration settings. To retrieve the id of the project owning team (owningTeam) use GET/auth/teams. To change the projects preset and configuration settings once the project has been created use POST/sast/scanSettings.

Usage

1. GET /auth/teams and get details of all teams
2. POST /projects and create new project with default configuration settings
3. POST /sast/scanSettings and define the scan settings

URL

http://localhost/cxrestapi/projects

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=2.0
cxOrigin: {request_origin}

Parameters

Required:

Project=[body] – Project details
name=[string] – Specifies the name of the project
owningTeam=[string] – Specifies the id of the team that owns the project
isPublic=[boolean] – Specifies whether the project is public or not

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
"name": "Project 5", \
"owningTeam": 1, \
"isPublic": true \
}' 'http://localhost/cxrestapi/projects'
```

Sample Response

```
{
  "id": 43,
  "link": {
    "rel": "self",
    "uri": "/projects/43"
  }
}
```

Success Response

Code: 201 Created

Error Response

Code: 400 Bad Request

Notes

Creates a new project with default preset and configuration settings. This creates a new project without a scan. If the request fails, it returns an error response.

Get Project Details by Id - GET /projects/{id} (v8.8.0 and up)

Get details of a specified project. To first retrieve the specific project id (id) use GET /projects.

Usage

1. GET /projects and get details of all visible projects
2. GET /projects/{id} and get details of a specified project

URL

http://localhost/cxrestapi/projects{id}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=2.0

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/projects/1'

Sample Response


```
{
  "id": 5,
  "teamId": 1,
  "name": "Project_4",
  "isPublic": true,
  "customFields": [],
  "links": [
    {
      "rel": "self",
      "uri": "/projects/5"
    },
    {
      "rel": "teams",
      "uri": "/auth/teams/"
    },
    {
      "rel": "latestscan",
      "uri": "/sast/scans?projectId=5&last=1"
    },
    {
      "rel": "allscans",
      "uri": "/sast/scans?projectId=5"
    },
    {
      "rel": "scansettings",
      "uri": "/sast/scanSettings/5"
    },
    {
      "type": "local",
      "rel": "source",
      "uri": null
    }
  ]
}
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves details of a specified project. If the request fails, it returns an error response.

Update Project by Id - PUT /projects/{id}

Update an existing project's details by project Id.

Usage

1. GET /projects/{id and get details of a specified project
2. PUT /projects/{id} and update the project

URL

http://localhost/cxrestapi/projects/{id}

Method

PUT

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=2.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

project=[body] – Specifies the project details:
name=[string] – Specifies the name of the project
owningTeam=[string] – Specifies the Id of the team that owns the project
customFields=[body] – specifies the custom field details:
id=[integer] – Unique Id of the custom field
value=[string] – custom field value (e.g. field label)

Curl Sample:

```
curl -X PUT --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "name": "Project 6", \
   "owningTeam": 1, \
   "customFields": [ \
     { \
       "id": 0, \
       "value": "string" \
     } \
   ] \
 }' 'http://localhost/cxrestapi/projects/1'
```

Sample Response

no content

Success Response

Code: 204 Accepted

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Updates an existing project's details by project Id. If the request fails, it returns an error response.

Update Project Name/Team Id - PATCH /projects/{id}

Update an existing project's name or team Id according to the Project Id.

Usage

1. GET /projects/{id} and get details of a specified project
2. PATCH /projects/{id} and update the project's name or team Id

URL

http://localhost/cxrestapi/projects/{id}

Method

PATCH

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=2.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

Project=[body] - Project details:
name=[string] – Specifies the name of the project
owningTeam=[string] – Specifies the Id of the team that owns the project

Curl Sample:

```
curl -X PATCH --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json;v=1.0' --header 'Authorization: Bearer
<access token> -d '{ \
"name": "Project 6", \
"owningTeam": 1 \
}' 'http://localhost/cxrestapi/projects/1'
```

Sample Response

no content

Success Response

Code: 204 Accepted

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Updates an existing project's name or team Id according to Project Id. The same team cannot have two projects with identical names. If the request fails, it returns an error response.

Delete Project by Id - DELETE /projects/{id}

Delete a specific project with all related scans according to Project Id.

Usage

1. GET /projects and get details of all projects
2. DELETE /projects/{id} and delete project by Id

URL

http://localhost/cxrestapi/projects/{id}

Method

DELETE

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=2.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

deleteProjectDto=[body] - A set of rules that specifies how the project should be deleted:
deleteRunningScans=[boolean] – Specifies whether running scans are to be deleted. Options are false/true. Default=False, if not specified.

Curl Sample:

```
curl -X DELETE --header 'Content-Type: application/json;v=1.0' --
header 'Accept:
application/json;v=1.0' --header 'Authorization: Bearer <access token>
-d '{ \
"deleteRunningScans": true \
}' 'http://localhost/cxrestapi/projects/1'
```

Sample Response

no content

Success Response

Code: 202 Accepted

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Deletes a specific project with all related scans according to Project Id. If the request fails, it returns an error response.

Create Branched Project - POST /projects/{id}/branch (v8.8.0 and up)

Create a branch of an existing project.

Usage

1. GET projects and get details of all projects
2. POST projects/{id}/branch

URL

http://localhost/cxrestapi/projects/{id}/branch

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project
project=[body] – Branched Project details:
name=[string] – Specifies the name of the branched project

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "name": "Project 6.1" \
 }' 'http://localhost/cxrestapi/projects/1/branch'
```

Sample Response

```
{
  "id": 5,
  "link": {
    "rel": "self",
    "uri": "/projects/5"
  }
}
```

Success Response

Code: 201 Created

Error Response

Code: 400 Bad Request

Notes

Creates a branch of an existing project. If the request fails, it returns an error response.

Get All Issue Tracking Systems - GET /issueTrackingSystems

Get details of all issue tracking systems (e.g. Jira) currently registered to CxSAST.

Usage

1. GET /issueTrackingSystems and get all issue tracking systems
2. POST /projects/{projectId}/issueTrackingSettings/jira and set issue tracking system as Jira

URL

http://localhost/cxrestapi/issueTrackingSystems

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

None

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/issueTrackingSystems'

Sample Response

```
[
   {
     "id": 2,
     "name": "Jira_Bug_Tracker_2",
     "type": "Jira",
     "url": http://xx.xx.x.xx:8080
   },
   {
     "id": 3,
     "name": "Jira_Bug_Tracker_3",
     "type": "Jira",
     "url": http://xx.xx.x.xx:8080
   }
 ]
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves details of all issue tracking systems (e.g. Jira) currently registered to CxSAST/CxOSA. If the request fails, it returns an error response.

Get Issue Tracking System Details by Id - GET /issueTrackingSystems/{id}/metadata

Get metadata for a specific issue tracking system (e.g. Jira) according to the Issue Tracking System Id.

Usage

1. GET /issueTrackingSystems and get details of all issue tracking systems
2. GET /issueTrackingSystems/{id}/metadata and get details of a specific issue tracking system by Id

URL

http://localhost/cxrestapi/issueTrackingSystems/{id}/metadata

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the issue tracking system

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>

'http://localhost/cxrestapi/issueTrackingSystems/1/metadata'

Sample Response


```
{
   "projects": [
     {
       "id": "10000",
       "name": "Example project",
       "issueTypes": [
         {
           "subtask": false,
           "id": "1",
           "name": "Example issue",
           "fields": [
             {
               "id": "1",
               "name": "Example field 1",
               "multiple": false,
               "required": false,
               "supported": true,
               "allowedValues": [
                 {
                   "id": "1",
                   "name": "v1"
                 },
                 {
                   "id": "2",
                   "name": "v2"
                 },
                 {
                   "id": "3",
                   "name": "v3"
                 }
               ]
             },
           ]
         }
       ]
     }
   ]
 }
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves metadata for a specific issue tracking system (e.g. Jira) according to the Issue Tracking System Id. If the request fails, it returns an error response.

Get Project Exclude Settings by Project Id - GET /projects/{id}/sourceCode/excludeSettings

Get details of a project's exclude folders/files settings according to the project Id.

Usage

1. GET /projects and get details of all projects
2. GET /projects/{id}/sourceCode/excludeSettings and get project exclude settings by project Id
3. PUT /projects/{id}/sourceCode/excludeSettings and set project exclude settings by project Id

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/excludeSettings

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[string] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer < access token>
'http://localhost/cxrestapi/projects/1/sourceCode/excludeSettings'

Sample Response


```
{
  "projectId": 41,
  "excludeFoldersPattern": "add-ons,connectors,doc,src,lib",
  "excludeFilesPattern":
"cvc3.js,spass.js,z3.js,readme.txt,smt_solver.js,readme.txt,find_sql_i
njections.js,jquery.js,logic.js",
  "link": {
    "rel": "self",
    "uri": "/projects/41/sourceCode/excludeSettings"
  }
}
```


Success Response

Code: 200 OK

Error Response

Code: 404 Not Found

Notes

Retrieves details of a project's exclude folders/files settings according to the project Id. If the request fails, it returns an error response.

Set Project Exclude Settings by Project Id - PUT /projects/{id}/sourceCode/excludeSettings

Set a project's exclude folders/files settings according to the project Id.

Usage

1. GET /projects/{id}/sourceCode/excludeSettings and get project exclude settings by project Id
2. PUT /projects/{id}/sourceCode/excludeSettings and set project exclude settings by project Id
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/excludeSettings

Method

PUT

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[string] – Unique Id of the project

Optional:

excludeSettings=[body] – Exclude folders/files pattern settings:
excludeFoldersPattern=[string] – comma separated list of folders, including wildcard patterns to exclude (e.g. add-ons, connectors, doc, src, lib)
excludeFilesPattern=[string] – comma separated list of files, including wildcard patterns to exclude (e.g. cvc3.js, spass.js, z3.js, readme.txt, smt_solver.js, readme.txt, find_sql_injections.js, jquery.js, logic.js)

Curl Sample:

```
curl -X PUT --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "excludeFoldersPattern": "add-ons, connectors, doc, src, lib", \
   "excludeFilesPattern": "cvc3.js, spass.js, z3.js, readme.txt,
smt_solver.js, readme.txt, find_sql_injections.js, jquery.js,
logic.js" \
 }' 'http://localhost/cxrestapi/projects/1/sourceCode/excludeSettings'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 404 Not Found

Notes

Defines a project's exclude folders/files settings according to the project Id. If the request fails, it returns an error response.

Get Remote Source Settings for GIT by Project Id - GET /projects/{id}/sourceCode/remoteSettings/git

Get a specific project's remote source settings for a GIT repository according to the Project Id.

Usage

1. GET /projects and get details of all projects
2. GET /projects/{id}/sourceCode/remoteSettings/git and get remote source settings for GIT by Project Id
3. POST /projects/{id}/sourceCode/remoteSettings/git and set remote source setting to GIT by Project Id

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/git

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/git'

Sample Response

```
{
      "url":
"https://xxxxx:xxxxx@github.com/dortal/DescriptionsTest.git",
      "branch": "refs/heads/master",
      "useSsh": false,
      "link": {
          "rel": "self",
          "uri": "/projects/2/sourceCode/remoteSettings/git"
      }
 }
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves a specific project's remote source settings for a GIT repository according to the Project Id. If the request fails, it returns an error response.

Set Remote Source Setting to GIT - POST /projects/{id}/sourceCode/remoteSettings/git

Set a specific project's remote source location to a GIT repository using SSH protocol. You can set the remote source location setting to GIT for an existing project (id) or you can first create a new project. To create a new project use POST /projects.

- The GIT Client needs to be configured and working on the CxManager for these API requests to work.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/git and set remote source setting to GIT
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/git

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: None
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

gitSettings=[body] – GIT settings details:
url=[string] – The url which is used to connect to the GIT repository (e.g. git@github.com:test/repo.git)
branch=[string] – The branch of a GIT repository (e.g. refs/heads/master)
privateKey=[string] – The private key (optional) which is used to connect to the GIT repository using SSH protocol
(e.g. -----BEGIN RSA PRIVATE KEY-----
MIIJKgIBAAKCAgEahM6IR0lb4Rag4s5JM+xyEfKiUotGlHx
SkeRjzXyWwjX5dAfR3K7pzHzn0rSMN7yUYlhZDLKff6R
-----END RSA PRIVATE KEY-----)

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
"url": "git@github.com:test/repo.git", \
"branch": "refs/heads/master", \
"privateKey": "-----BEGIN RSA PRIVATE KEY-----
MIIJKgIBAAKCAgEahM6IR0lb4Rag4s5JM+xyEfKiUotGlHx
SkeRjzXyWwjX5dAfR3K7pzHzn0rSMN7yUYlhZDLKff6R \
-----END RSA PRIVATE KEY----- \
" \
}'
'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/git'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Defines a specific project's remote source location to a GIT repository using SSH protocol. If the request fails, it returns an error response.

The url can contain a different method of authentication when not using the optional privateKey, for example:

- Private repository with credentials: url - https://<username>:<password>@github.com/test/repo.git
- Private repository with token: url – https://<token>@github.com/test/repo.git
- Public repository (no credentials): url - https://github.com/test/repo.git

It is also possible to provide an SSH certificate instead of using a privateKey. To set the project's remote source location to a GIT repository using a SSH certificate use POST /projects/{id}/sourceCode/remoteSettings/git/ssh.

Get Remote Source Settings for SVN by Project Id - GET /projects/{id}/sourceCode/remoteSettings/svn

Get a specific project's remote source location settings for SVN repository according to the Project Id.

Usage

1. GET /projects and get details of all projects
2. GET /projects/{id}/sourceCode/remoteSettings/svn and get remote source settings for SVN by Project Id
3. POST /projects/{id}/sourceCode/remoteSettings/svn and SET remote source settings for SVN by Project Id

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/svn

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/svn'

Sample Response

```
{
      "uri": {
          "absoluteUrl": "svn://xx.xx.x.xxx/testrepo",
          "port": 3690
      },
      "paths": [
          "/trunk/BS_CSharp/BookStore_NET_17588_lines",
          "/LongPath-JS_XSS_DOM"
      ],
      "useSsh": false,
      "link": {
          "rel": "self",
          "uri": "/projects/2/sourceCode/remoteSettings/svn"
      }
 }
```

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves a specific project's remote source location settings for SVN repository according to the Project Id. If the request fails, it returns an error response.

Set Remote Source Setting to SVN - POST /projects/{id}/sourceCode/remoteSettings/svn

Set a specific project's remote source location to a SVN repository using SSH protocol. You can set the remote source location setting to SVN for an existing project (id) or you can first create a new project. To create a new project use POST /projects.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/svn and set remote source setting to SVN
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/svn

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

svnSettings=[body] – SVN settings details:
uri – Uri of SVN repository
absoluteUrl=[string] – Specifies the absolute url (e.g. http://<server_ip>/svn/testrepo)
port=[integer] – Specifies the port number of the uri (e.g. 8080)
paths=[string] – Specifies the list of paths to scan at SVN repository (e.g. /trunk)
credentials – Specifies credentials for password-based authentication against the SVN repository.
username=[string]
password=[string]
privateKey=[string] – The private key (optional) which is used to connect to the SVN repository using SSH protocol
(e.g. -----BEGIN RSA PRIVATE KEY-----
MIIJKgIBAAKCAgEahM6IR0lb4Rag4s5JM+xyEfKiUotGlHx
SkeRjzXyWwjX5dAfR3K7pzHzn0rSMN7yUYlhZDLKff6R
-----END RSA PRIVATE KEY-----)

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "uri": { \
     "absoluteUrl": "http://<server_ip>/svn/testrepo", \
     "port": 8080 \
   }, \
   "paths": [ \
     "/trunk" \
   ], \
   "credentials": { \
     "userName": "<username>", \
     "password": "<password>" \
   }, \
   "privateKey": "-----BEGIN RSA PRIVATE KEY----
-
MIIJKgIBAAKCAgEahM6IR0lb4Rag4s5JM+xyEfKiUotGlHx

SkeRjzXyWwjX5dAfR3K7pzHzn0rSMN7yUYlhZDLKff6R \
                                    -----END RSA PRIVATE KEY----- \
 " \
 }'
'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/svn'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Defines a specific project's remote source location to a SVN repository using SSH protocol. If the request fails, it returns an error response.

The url can contain a different method of authentication when not using the optional privateKey.

- Private repository with credentials: url - https://<username>:<password><server_ip>/svn/testrepo
- Private repository with token: url – https://<token><server_ip>/svn/testrepo
- Public repository (no credentials): url - https://<server_ip>/svn/testrepo

It is also possible to provide an SSH certificate instead of using a Private Key. To set the project's remote source location to a SVN repository using a SSH certificate use POST /projects/{id}/sourceCode/remoteSettings/svn/ssh.

Get Remote Source Settings for TFS by Project Id - GET /projects/{id}/sourceCode/remoteSettings/tfs

Get a specific project's remote source location settings for TFS repository according to the Project Id.

Usage

1. GET /projects and get details of all projects
2. GET /projects/{id}/sourceCode/remoteSettings/tfs and get remote source settings for TFS by Project Id
3. POST /projects/{id}/sourceCode/remoteSettings/tfs and set remote source settings for TFS by Project Id

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/tfs

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/tfs'

Sample Response

```
{
      "uri": {
          "absoluteUrl": "http://tfsxxxx:8080/tfs/DefaultCollection",
          "port": 8080
      },
      "paths": [
          "/Test/Beijing_1232_lines"
      ],
      "link": {
          "rel": "self",
          "uri": "/projects/89/sourceCode/remoteSettings/tfs"
      }
 }
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves a specific project's remote source location settings for TFS repository according to the Project Id. If the request fails, it returns an error response.

Set Remote Source Setting to TFS - POST /projects/{id}/sourceCode/remoteSettings/tfs

Set a specific project's remote source location to a TFS repository. You can set the remote source location setting to TFS for an existing project (id) or you can first create a new project. To create a new project use POST /projects.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/tfs and set remote source setting to TFS
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/tfs

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

tfsSettings=[body] – TFS settings details:
credentials – Specifies credentials for password-based authentication against the TFS repository:
username=[string]
password=[string]
uri – Uri of TFS repository:
absoluteUrl=[string] – Specifies the absolute url (e.g. http://<site_name>/tfs/DefaultCollection)
port=[integer] – Specifies the port number of the uri (e.g. 8080)
paths=[string] – Specifies the list of paths to scan at TFS repository (e.g. /Root/Optimization/V6.2.2.9-branch/CSharp/Graph, /Root/test)

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "credentials": { \
     "userName": "<username>", \
     "password": "<password>" \
   }, \
   "uri": { \
     "absoluteUrl": "http://<site_name>/tfs/DefaultCollection", \
     "port": 8080 \
   }, \
   "paths": [ \
     "/Root/Optimization/V6.2.2.9-branch/CSharp/Graph, /Root/test" \
   ] \
 }'
'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/tfs'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Defines a specific project's remote source location to a TFS repository. If the request fails, it returns an error response. The setting of the remote source location to TFS is performed before creating a new SAST scan. To create a new SAST scan use

Get Remote Source Settings for Custom by Project Id - GET /projects/{id}/sourceCode/remoteSettings/custom

Get a specific project's remote source location settings for custom repository (e.g. source pulling) according to the Project Id.

Usage

1. GET /projects and get details of all projects
2. GET /projects/{id}/sourceCode/remoteSettings/custom and get remote source settings for Custom repository by Project Id
3. POST /projects/{id}/sourceCode/remoteSettings/custom and set remote source settings for Custom repository by Project Id

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/custom

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/custom'

Sample Response

```
{
     "path":
"\\\\storage\\QA\\Projects_new\\Java\\1_Under_70k\\BookStore_Small_CLI
",
     "pullingCommandId": 1,
     "link": {
         "rel": "self",
         "uri": "/projects/2/sourceCode/remoteSettings/custom"
     }
 }
```

Success Response

Code: 200 OK

Error Response

Code: 404 Not Found

Notes

Retrieves specific project's remote source location settings for custom repository (e.g. source pulling) according to the Project Id. If the request fails, it returns an error response.

Set Remote Source Setting for Custom by Project Id - POST /projects/{id}/sourceCode/remoteSettings/custom

Set a specific project's remote source location settings for custom repository (e.g. source pulling) according to the Project Id.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/custom and set remote source setting to custom repository
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/custom

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

customSourceSettings=[body] - Custom settings details:
Path=[string] – Path to the network folders containing the project code
preScanCommandId=[integer] – Unique Id of script that pulls the source code
credentials=[body] – Specifies credentials to access the network:
username=[string]
password=[string]

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
code> -d '{ \
   "path":
"\\\\storage\\QA\\Projects_new\\Java\\1_Under_70k\\BookStore_Small_CLI
", \
     "pullingCommandId: 1", \
   "preScanCommandId": 1, \
   "credentials": { \
     "userName": "<username>", \
     "password": "<password>" \
   } \
 }' 'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/custom'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request

Notes

Defines a specific project's remote source location settings for custom repository (e.g. source pulling) according to the Project Id. If the request fails, it returns an error response.

Get Remote Source Settings for Shared by Project Id - GET /projects/{id}/sourceCode/remoteSettings/shared (v8.7.0 and up)

Get a specific project's remote source location settings for shared repository according to the Project Id.

Usage

1. GET /projects and get details of all projects
2. GET /projects/{id}/sourceCode/remoteSettings/shared and get remote source settings for shared repository by Project Id
3. POST /projects/{id}/sourceCode/remoteSettings/shared and set remote source settings for shared repository by Project Id

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/shared

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/shared'

Sample Response

```
{
   "paths": [
     \\\\storage\\qa\\projects_new\\CPP\\1_Under_70k\\cpp_22_LOC
   ],
   "link": {
     "rel": "self",
     "uri": "/projects/2/sourceCode/remoteSettings/shared"
   }
 }
```

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves a specific project's remote source location settings for shared repository according to the Project Id. If the request fails, it returns an error response.

Set Remote Source Setting to Shared - POST /projects/{id}/sourceCode/remoteSettings/shared

Set a specific project's remote source location to a shared repository. You can set the remote source location setting to shared for an existing project (id) or you can first create a new project. To create a new project use POST /projects.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/shared and set remote source setting to Shared
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/shared

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

sharedSettings=[body] – Shared settings details:
paths=[string] – Specifies the list of paths to scan at the shared repository (e.g. \\\\storage\\qa\\projects_new\\CPP\\1_Under_70k\\cpp_22_LOC)
Credentials – Specifies credentials for password-based authentication against the shared repository.
username=[string]
password=[string]

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "paths":
[  "\\\\storage\\qa\\projects_new\\CPP\\1_Under_70k\\cpp_22_LOC" \
   ], \
   "credentials": { \
     "userName": "<username>", \
     "password": "<password>" \
   } \
 }'
'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/share
d'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request

Notes

Defines a specific project's remote source location to a shared repository. If the request fails, it returns an error response. The setting of the remote source location to shared should be performed before creating a new SAST scan. To create a new SAST scan use POST /sast/scans.

Get Remote Source Settings for Perforce by Project Id - GET /projects/{id}/sourceCode/remoteSettings/perforce

Get a specific project's remote source location settings for Perforce repository according to the Project Id.

Usage

1. GET /projects and get details of all projects
2. GET /projects/{id}/sourceCode/remoteSettings/perforce and get remote source settings for Perforce repository by Project Id
3. POST /projects/{id}/sourceCode/remoteSettings/perforce and set remote source settings for Perforce repository by Project Id

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/perforce

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the specific project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/perforce'

Sample Response

```
{
      "uri": {
          "absoluteUrl": "xx.xx.x.xxx",
          "port": 1666
      },
      "paths": [
          "//Depot_1"
      ],
      "browseMode": "Depot",
      "link": {
          "rel": "self",
          "uri": "/projects/89/sourceCode/remoteSettings/perforce"
      }
 }
```

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves a specific project's remote source location settings for Perforce repository according to the Project Id. If the request fails, it returns an error response.

Set Remote Source Setting to Perforce - POST /projects/{id}/sourceCode/remoteSettings/perforce

Set a specific project's remote source location to a Perforce repository. You can set the remote source location setting to Perforce for an existing project (id) or you can first create a new project. To create a new project use POST /projects.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/perforce and set remote source setting to Perforce
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/perforce

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the specific project

perforceSettings=[body] – Perforce settings details:
Credentials – Specifies credentials for password-based authentication against the Perforce repository:
username=[string]
password=[string]
uri – Specifies the uri of Perforce repository:
absoluteUrl=[string] – Specifies the absolute url (e.g. <server_ip>)
port=[integer] – Specifies the port number of this Uri (e.g. 8080)
paths=[string] – Specifies the list of paths to scan at Perforce repository (e.g. ////depot)
browseMode=[string] – Specifies the browsing mode of the Perforce repository (depot for shared or workspace for grouped).

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "credentials": { \
     "userName": "<username>", \
     "password": "<password>" \
   }, \
   "uri": { \
     "absoluteUrl": "<server_ip>", \
     "port": 8080 \
   }, \
   "paths": [ \
     "////depot" \
   ], \
   "browseMode": "Depot" \
 }'
'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/perfo
rce'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request

Notes

Defines a specific project's remote source location to a Perforce repository. If the request fails, it returns an error response. The setting of the remote source location to Perforce should be performed before creating a new SAST scan. To create a new SAST scan use POST /sast/scans.

Set Remote Source Setting to GIT using SSH - POST /projects/{id}/sourceCode/remoteSettings/git/ssh

Set a specific project's remote source location to a GIT repository using the SSH protocol (i.e. SSH Certificate). You can set the remote source location setting to GIT for an existing project (id) or you can first create a new project. To create a new project use POST/projects.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/git/ssh and set remote source setting to GIT using the SSH protocol
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/git/ssh

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project
url=[string] – The URL which is used to connect to the GIT repository (e.g. git@github.com:test_repo/test.git)
branch=[string] – The branch of a GIT repository (e.g. refs/heads/master)
privatekey=[file] – The SSH certificate which is used to connect to the GIT repository using SSH protocol (multipart/form-data)

Curl Sample:

curl -X POST --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' --header 'Authorization: Bearer <access token> -F url=git%40github.com%3Atest%2Frepo.git -F branch=refs%2Fheads%2Fmaster 'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/git/ssh'

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request

Notes

Defines a specific project's remote source location to a GIT repository using the SSH protocol (i.e. SSH Certificate). If the request fails, it returns an error response. It is also possible to provide a private key instead of using a SSH certificate. To set the project's remote source location to a GIT repository using a private key use POST /projects/{id}/sourceCode/remoteSettings/git.

Set Remote Source Setting to SVN using SSH - POST /projects/{id}/sourceCode/remoteSettings/svn/ssh

Set a specific project's remote source location to a SVN repository which uses the SSH protocol (i.e. SSH Certificate). You can set the remote source location setting to SVN for an existing project (id) or you can first create a new project. To create a new project use POST/projects.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/remoteSettings/svn/ssh and set remote source setting to SVN using the SSH protocol
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/remoteSettings/svn/ssh

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the specific project
absoluteUrl=[string] – The URL which is used to connect to the SVN repository (e.g. http://<server_ip>/svn/testrepo)
port=[integer] – Specifies the port number of SVN repository url
paths=[string] – Specifies the paths of the SVN repository (e.g. /trunk)
privatekey=[file] – The SSH certificate which is used to connect to the SVN repository using SSH protocol (multipart/form-data)

Curl Sample

```
curl -X POST --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' --header 'Authorization: Bearer <access token> -F absoluteUrl=http%3A%2F%2F%3Cserver_ip%3E%2Fsvn%2Ftestrepo -F port=8080 -F paths=%2Ftrunk  'http://localhost/cxrestapi/projects/1/sourceCode/remoteSettings/svn/ssh'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request

Notes

Defines a specific project's remote source location to a SVN repository which uses the SSH protocol (i.e. SSH Certificate). If the request fails, it returns an error response. It is also possible to provide a private key instead of using a SSH certificate. To set the project's remote source location to a SVN repository using a private key use POST /projects/{id}/sourceCode/remoteSettings/svn.

Upload Source Code Zip File - POST /projects/{id}/sourceCode/attachments (v8.6.0 and up)

Upload a zip file that contains the source code for scanning. You can upload a zip file to an existing project or you can first create a new project and then upload the file. To create a new project use POST /projects. The upload of a zip file is performed before creating a new SAST scan. To create a new SAST scan use POST /sast/scans.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/sourceCode/attachments and upload the zipped source code
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/sourceCode/attachments

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] - Unique Id of the project

Content-type=[multipart/form-data]:
zippedSource=[file] – Zipped source code file to scan (form-data)

Curl Sample:

curl -X POST "http(s)://<CX_HOST>/cxrestapi/projects/<PROJECT_ID>/sourceCode/attachments" -H "Accept: application/json" -H "Authorization: Bearer <ACCESS_TOKEN>" -H "Content-Type: multipart/form-data" -F zippedSource=<ZIP_FILE_PATH>

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request

Notes

Uploads a zip file that contains the source code for scanning. If the request fails, it returns an error response.

Get All Custom Tasks - GET /customTasks

Get details of all custom tasks (e.g. pre/post scan actions).

Usage

1. GET /customTasks and get details of all custom tasks
2. GET /customTasks/{customTaskId} and get details according to custom task Id

URL

http://localhost/cxrestapi/customTasks

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

None

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/customTasks'

Sample Response


```
[
  {
    "id": 6,
    "name": "pre2",
    "type": "SOURCE_CONTROL_COMMAND",
    "data": "copyFiles.bat",
    "link": {
      "rel": "self",
      "uri": "/customTasks/6"
    }
  },
  {
    "id": 2,
    "name": "post",
    "type": "POST_SCAN_COMMAND",
    "data": "preScanAction.bat <CSV_output>",
    "link": {
      "rel": "self",
      "uri": "/customTasks/2"
    }
  }
]
```


Success Response

Code: 200 OK

Error Response

Code: 404 Not Found

Notes

Retrieves details of all custom tasks. If the request fails, it returns an error response.

Get Custom Task by Id - GET /customTasks/{id}

1. Get details of a specific custom task (e.g. pre/post scan actions) according to the custom task Id.GET /customTasks and get details of all custom tasks
2. GET /customTasks/{id} and get details according to custom task Id

URL

http://localhost/cxrestapi/customTasks/{id}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
|Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the custom task

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/customTasks/1'

Sample Response


```
{
  "id": 6,
  "name": "pre2",
  "type": "SOURCE_CONTROL_COMMAND",
  "data": "copyFiles.bat",
  "link": {
    "rel": "self",
    "uri": "/customTasks/6"
  }
}
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves details of a specific custom task (e.g. pre/post scan actions) according to the custom task Id. If the request fails, it returns an error response.

Get All Custom Fields - GET /customFields

Get details of all custom fields.

Usage

GET /customFields and get details of all custom fields

URL

http://localhost/cxrestapi/customFields

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

None

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/customFields'

Sample Response

```
[
  {
    "id": 1,
    "name": "Custom Field"
  }
]
```

Success Response

Code: 200 OK

Error Response

Code: 404 Not Found

Notes

Retrieves details of all custom fields. If the request fails, it returns an error response.

Set Data Retention Settings by Project Id - POST /projects/{id}/dataRetentionSettings

Set the data retention settings according to Project Id.

Usage

1. POST /projects and create new project with default configuration settings
2. POST /projects/{id}/dataRetentionSettings and set the data retention settings according to Project Id.
3. POST /sast/scans and create a new scan

URL

http://localhost/cxrestapi/projects/{id}/dataRetentionSettings

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project
dataRetentionSettings=[body] – Data retention settings
scansToKeep=[integer] – The amount of scans to keep before they are deleted (1-1000 or null)

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "scansToKeep": 10 \
 }' 'http://localhost/cxrestapi/projects/1/dataRetentionSettings'
```

Sample Response

no content

Success Response

Code: 204 No Content

Error Response

Code: 400 Bad Request

Notes

Defines the data retention settings according to Project Id. If the request fails, it returns an error response.

Set Issue Tracking System as Jira by Id - POST /projects/{id}/issueTrackingSettings/jira

Set a specific issue tracking system as Jira according to Project Id.

Usage

1. GET /issueTrackingSystems and get all issue tracking systems
2. POST /projects/{id}/issueTrackingSettings/jira and set issue tracking system as Jira

URL

http://localhost/cxrestapi/projects/{id}/issueTrackingSettings/jira

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

jiraSettings=[body] – Jira issue tracking details:
issueTrackingSystemId=[integer] – Specifies the issue tracking system Id
jiraProjectId=[string] – Specifies the specific Id of Jira project
issueType – Specifies the Jira project's issue type:
id=[string] – Specifies the Id of issue type
fields=[Array] - Specifies the list of fields associated with the issue type
id=[string] – Specifies the Id of the field
values=[string] – Specifies single or multiple values

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "issueTrackingSystemId": 1, \
   "jiraProjectId": "12901", \
   "issueType": { \
     "id": "123", \
     "fields": [ \
       { \
         "id": "priority", \
         "values": [ \
           "1" \
         ] \
       }, \
       { \
         "id": "customfield_1", \
```

```
        "values": [ \
          "c1", \
          "c2" \
        ] \
      } \
    ] \
  } \
}' 'http://localhost/cxrestapi/projects/1/issueTrackingSettings/jira'
```

Sample Response

```
{
   "projects": [
     {
       "id": "10000",
       "name": "Example project",
       "issueTypes": [
         {
           "subtask": false,
           "id": "1",
           "name": "Example issue",
           "fields": [
             {
               "id": "1",
               "name": "Example field 1",
               "multiple": false,
               "required": false,
               "supported": true,
               "allowedValues": [
                 {
                   "id": "1",
                   "name": "v1"
                 },
                 {
                   "id": "2",
                   "name": "v2"
                 },
                 {
                   "id": "3",
                   "name": "v3"
                 }
               ]
             },
           ]
         }
       ]
     }
   ]
}
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Defines a specific issue tracking system as Jira according to Project Id. If the request fails, it returns an error response.

Get All Preset Details - GET /sast/presets

Get details of all presets. The retrieved preset id (id) can be used to get details of a specific preset using GET /presets/sast/{id}.

Usage

1. GET /sast/presets and gets details of all presets
2. GET /sast/presets/{id} and get details of a specific preset

URL

http://localhost/cxrestapi/sast/presets

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

None

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/presets'

Sample Response

```
[
    {
      "id": 36,
      "name": "Checkmarx Default",
      "ownerName": "CxUser",
      "link": {
        "rel": "self",
        "uri": "/sast/presets/36"
      }
    },
    {
      "id": 1,
      "name": "All",
      "ownerName": "CxUser",
      "link": {
        "rel": "self",
        "uri": "/sast/presets/1"
      }
    }
 ]
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves details of all presets. If the request fails, it returns an error response.

Get Preset Details by Preset Id - GET /sast/presets/{id}

Get details of a specified preset by Id. In order to retrieve the preset Id (id) of a specific preset you should first retrieve the details of all presets. To retrieve the details of all presets use GET /sast/presets.

Usage

1. GET /sast/presets and get details of all presets
2. GET /sast/presets/{id} and get details of a specific preset

URL

http://localhost/cxrestapi/sast/presets{id}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[string] – Unique Id of the preset

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/presets/1'

Sample Response

```
{
   "queryIds": [
     51,
     52,
     55,
     56,
     57,
     132,
     133,
     134,
     135,
     ...
   ],
   "id": 36,
   "name": "Checkmarx Default",
   "ownerName": "CxUser",
   "link": {
     "rel": "self",
     "uri": "/sast/presets/36"
   }
 }
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves details of a specified preset by Id. The response includes all the query Ids (queryId) associated with the specified preset. If the request fails, it returns an error response.

CxSAST (REST) API - SAST Scans

This section covers CxREST APIs for working with scan tasks.

Get All Scans for Project - GET /sast/scans

Get details of all SAST scans for a specific project.

Usage

1. GET /projects and get details of all projects
2. GET /sast/scans and get details of all scans for a specific project

URL

http://localhost/cxrestapi/sast/scans

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Optional:

projectId=[string] – Unique Id of the project
scanStatus=[string] – The current status of the scan (1="New", 2="PreScan", 3="Queued", 4="Scanning", 6="PostScan", 7="Finished", 8="Canceled", 9="Failed", 10="SourcePullingAndDeployment", 1001="None").
last=[integer] – Number of last scans to include.

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/scans?last=3&scanStatus=Finished&projectId=1'

Sample Response


```
[
  {
    "id": 1000000,
    "project": {
      "id": 1,
      "name": "Project 1 (CxTechDocs)",
      "link": null
    },
    "status": {
      "id": 7,
      "name": "Finished",
      "details": {
        "stage": "",
        "step": ""
      }
    },
    "scanType": {
      "id": 1,
      "value": "Regular"
    },
    "comment": "",
    "dateAndTime": {
      "startedOn": "2018-06-18T00:59:15.407",
      "finishedOn": "2018-06-18T01:09:12.707",
      "engineStartedOn": "2018-06-18T00:59:15.407",
      "engineFinishedOn": "2018-06-18T01:09:11.443"
    },
    "resultsStatistics": {
      "link": null
    },
    "scanState": {
      "path": " N/A (Zip File)",
      "sourceId": "0000000189_001556351823_00-432903868",
      "filesCount": 189,
      "linesOfCode": 33594,
      "failedLinesOfCode": 0,
      "cxVersion": "8.8.0.1351",
      "languageStateCollection": [
        {
          "languageID": 1073741824,
          "languageName": "Common",
          "languageHash": "1168085659085622",
          "stateCreationDate": "2018-06-17T07:30:33.443"
        },
        {
          "languageID": 2,
          "languageName": "Java",
          "languageHash": "6487237970207034",
          "stateCreationDate": "2018-06-17T07:30:33.443"
```

```
        },
        {
          "languageID": 8,
          "languageName": "JavaScript",
          "languageHash": "3602822811217894",
          "stateCreationDate": "2018-06-17T07:30:33.443"
        },
        {
          "languageID": 262144,
          "languageName": "Typescript",
          "languageHash": "1939975091058023",
          "stateCreationDate": "2018-06-17T07:30:33.443"
        },
        {
          "languageID": 64,
          "languageName": "VbScript",
          "languageHash": "1349101913133594",
          "stateCreationDate": "2018-06-17T07:30:33.443"
        }
      ]
    },
    "owner": "admin@cx",
    "origin": "Web Portal",
    "initiatorName": "admin admin",
    "owningTeamId": "00000000-1111-1111-b111-989c9070eb11",
    "isPublic": true,
    "isLocked": false,
    "isIncremental": false,
    "scanRisk": 100,
    "scanRiskSeverity": 100,
    "engineServer": {
      "id": 1,
      "name": "Localhost",
      "link": null
    },
    "finishedScanStatus": {
      "id": 0,
      "value": "None"
    },
    "partialScanReasons": null
  }
]
```


Success Response

Code: 200 OK

Error Response

Code: 400 Not Found

Notes

Retrieves details of all SAST scans for a specific project. If the request fails, it returns an error response.

Create New Scan – POST /sast/scans

Create a new SAST scan and assign it to a project. When initiating a scan you can send a media type header (cxOrigin) which indicates which client is being used to send the scan request (e.g. Jenkins, Bamboo, TeamCity, Maven, etc.). If not defined, default is Other.

- The 'SAVE-PROJECT' permission is required to execute this API.

Usage

1. POST /sast/scanSettings and update the SAST scan preset and configuration settings
2. POST /sast/scans and create a new SAST scan.

URL

http://localhost/cxrestapi/sast/scans

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

scan=[body] – Scan details:
projectId=[integer] – Unique Id of the project to be scanned
isIncremental=[boolean] – Specifies whether the requested scan is incremental or full scan
isPublic=[boolean] – Specifies whether the requested scan is public or private
forceScan=[boolean] – Specifies whether the code should be scanned or not, regardless of whether changes were made to the code since the last scan.
comment=[string] – Specifies the scan comment.

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "projectId": 1, \
   "isIncremental": false, \
   "isPublic": true, \
   "forceScan": true, \
   "comment": "build#1" \
 }' 'http://localhost/cxrestapi/sast/scans'
```

Sample Response

```
{
  "id": 1000062,
  "link": {
    "rel": "self",
    "uri": "/sast/scans/1000062"
  }
}
```

Success Response

Code: 201 Created

Error Response

Code: 400 Bad Request

Notes

Creates a new SAST scan and assigns it to a project. If the request fails, it returns an error response. Must be a valid project in order to create a new scan.

Get SAST Scan Details by Scan Id - GET /sast/scans/{id}

This section includes REST APIs for retrieving scan details for CxSAST scans.

Get details of a specific SAST scan. Scan details can only be retrieved once a scan has been performed and the scan Id (id) is known. To create a new scan use POST /sast/scan.

Usage

1. POST /sast/scan and create a new scan
2. GET /sast/scans/{id} and get details of a specific scan

URL

http://localhost/cxrestapi/sast/scans/{id}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the scan

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/sast/scans/1000062'

Sample Response


```
{
  "id": 1000057,
  "project": {
    "id": 41,
    "name": "Project 1 (8.8)",
    "link": {
      "rel": "project",
      "uri": "/projects/41"
    }
  },
  "status": {
    "id": 7,
    "name": "Finished",
    "details": {
      "stage": "",
      "step": ""
    }
  },
```

```
"scanType": {
  "id": 1,
  "value": "Regular"
},
"comment": "",
"dateAndTime": {
  "startedOn": "2018-05-27T17:50:24.83",
  "finishedOn": "2018-05-27T17:52:28.71",
  "engineStartedOn": "2018-05-27T17:50:24.83",
  "engineFinishedOn": "2018-05-27T17:52:28.427"
},
"resultsStatistics": {
  "link": {
    "rel": "results-statistics",
    "uri": "/sast/scans/1000057/resultsStatistics"
  }
},
"scanState": {
  "path": " N/A (Zip File)",
  "sourceId": "0000000034_000315599833_001616083091",
  "filesCount": 34,
  "linesOfCode": 6864,
  "failedLinesOfCode": 6,
  "cxVersion": "8.8.0.838",
  "languageStateCollection": [
    {
      "languageID": 1073741824,
      "languageName": "Common",
      "languageHash": "1060095108086482",
      "stateCreationDate": "2018-05-10T10:39:11.26"
    },
    {
      "languageID": 1,
      "languageName": "CSharp",
      "languageHash": "2082253736012464",
      "stateCreationDate": "2018-05-21T14:27:30.48"
    },
    {
      "languageID": 8,
      "languageName": "JavaScript",
      "languageHash": "0551811326035748",
      "stateCreationDate": "2018-05-10T10:39:11.26"
    },
    {
      "languageID": 64,
      "languageName": "VbScript",
      "languageHash": "1349101913133594",
      "stateCreationDate": "2018-05-10T10:39:11.26"
    }
  ]
},
"owner": "admin@cx",
```

```
  "origin": "Web Portal",
  "initiatorName": "admin admin",
  "owningTeamId": "00000000-1111-1111-b111-989c9070eb11",
  "isPublic": true,
  "isLocked": false,
  "isIncremental": false,
  "scanRisk": 85,
  "scanRiskSeverity": 55,
  "engineServer": {
    "id": 4,
    "name": "100plus",
    "link": {
      "rel": "engine-server",
      "uri": "/sast/engineServers/4"
    }
  },
  "finishedScanStatus": {
    "id": 1,
    "value": "Completed"
  },
  "partialScanReasons": []
}
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves details of a specific SAST scan. Must be a valid SAST scan in order to retrieve scan details. Status of the scan can be: 1="New", 2="PreScan", 3="Queued", 4="Scanning", 6="PostScan", 7="Finished", 8="Canceled", 9="Failed", 10="SourcePullingAndDeployment", 1001="None". If the request fails, it returns an error response. Must be a valid SAST scan in order to retrieve scan details.

Add/Update a Comment by Scan Id - PATCH /sast/scans{id}

Add a new comment or update an existing comment according to the scan Id.

Usage

1. POST /sast/scans and create a new SAST scan.
2. PATCH /sast/scans/{id} and add a comment
3. GET /sast/scans/{id} and get details of a specific scan

URL

http://localhost/cxrestapi/sast/scans/{id}

Method

PATCH

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

Id=[integer] – Unique Id of a specific scan
updatedScanDto=[body] – comment details:
comment=[string] – Specifies the comment content

Curl Sample:

```
curl -X PATCH --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "comment": "Build#2" \
 }' 'http://localhost/cxrestapi/sast/scans/1000062
```

Sample Response

no content

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Adds a new comment or update an existing comment according to the scan Id. If the request fails, it returns an error response.

Delete Scan by Scan Id - DELETE /sast/scans/{id}

Delete specific SAST scan according to scan Id.

Usage

1. GET /sast/scans/{id} and get details of a specific scan
2. DELETE /sast/scans/{id} and delete a specific scan

URL

http://localhost/cxrestapi/sast/scans/{id}

Method

DELETE

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the scan

Curl Sample:

curl -X DELETE --header 'Accept: application/json' --header 'Authorization: Bearer <acess token>
'http://localhost/cxrestapi/sast/scans/1000062'

Sample Response

no content

Success Response

Code: 202 Accepted

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Deletes the specific SAST scan according to scan Id. If the request fails, it returns an error response. Must be a valid SAST scan in order to delete scan.

Get Statistic Results by Scan Id - GET /sast/scans/{id}/resultsStatistics (v8.8.0 and up)

Get statistic results for a specific scan.

Usage

GET /sast/scans/{id}/resultsStatistics and get statistic results for a specific scan

URL

http://localhost/cxrestapi/sast/scans/{id}/resultsStatistics

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[string] – Unique Id of the scan

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/scans/1000062/resultsStatistics'

Sample Response

```
{
  "highSeverity": 25,
  "mediumSeverity": 22,
  "lowSeverity": 80,
  "infoSeverity": 0,
  "statisticsCalculationDate": "2018-05-27T17:52:09.687"
}
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves statistic results for a specific scan. If the request fails, it returns an error response.

Update Queued Scan Status by Scan Id - PATCH /sast/scansQueue/{id}

Update (Cancel) a running scan in the queue according to the scan Id.

Usage

1. GET /sast/scansQueue and get all scan queue details
2. PATCH /sast/scansQueue/{id} and update queued scan status

URL

http://localhost/cxrestapi/sast/scansQueue/{id}

Method

PATCH

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

URL Parameters

Required:

id=[integer] – Unique Id of a specific scan in the queue

scanRequest=[body] – scan status:
status=[string] – Requested scan status in the queue. Status option includes Canceled.

Curl Sample:

```
curl -X PATCH --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d \
'{ \
"status": "Canceled" \
}' \
'http://localhost/cxrestapi/sast/scansQueue/1000062'
```

Sample Response

no content

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Updates a running scan in the queue according to the scan Id. If the request fails, it returns an error response.

Get All Scan Details in Queue - GET /sast/scansQueue

Get details of all SAST scans in the scans queue. The scan queue can also be filtered by project Id. In order to retrieve the details of a SAST scan in the scan queue you should first create a scan. To create a new scan use POST /sast/scans.

Usage

1. POST /sast/scans and create a new SAST scan
2. GET /sast/scansQueue and get details of all SAST scans in the scans queue

URL

http://localhost/cxrestapi/sast/scansQueue

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Optional:

projectId=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/scansQueue?projectId=1'

Sample Response


```
[
  {
    "id": 1000008,
    "stage": {
      "id": 7,
      "value": "Finished"
    },
    "stageDetails": "Scan completed",
    "stepDetails": null,
    "project": {
      "id": 32,
      "name": "Project 1",
      "link": {
        "rel": "project",
        "uri": "/projects/32"
      }
    },
    "engine": {
      "id": 1,
```

```
      "link": {
        "rel": "engine-server",
        "uri": "/sast/engineServers/1"
      }
    },
    "languages": [
      {
        "id": 1,
        "name": "CSharp"
      },
      {
        "id": 8,
        "name": "JavaScript"
      },
      {
        "id": 64,
        "name": "VbScript"
      },
      {
        "id": 1073741824,
        "name": "Common"
      }
    ],
    "teamId": "00000000-1111-1111-b111-989c9070eb11",
    "dateCreated": "2018-06-03T15:39:52.13",
    "queuedOn": "2018-06-03T15:40:06.973",
    "engineStartedOn": "2018-06-03T15:40:28.273",
    "completedOn": "2018-06-03T15:42:28.637",
    "loc": 6836,
    "isIncremental": false,
    "isPublic": true,
    "origin": "Other",
    "queuePosition": 1,
    "totalPercent": 99,
    "stagePercent": 100,
    "initiator": "admin admin"
  }
]
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves details of all SAST scans in the scans queue. If the request fails, it returns an error response. If the request fails, it returns an error response. If the queue is empty it does not provide any details. Must be a valid (registered) engine server in order to get details. Possible stages are: 1=New, 2=PreScan, 3=Queued, 4=Scanning, 6=PostScan, 7=Finished, 8=Canceled, 9=Failed, 10=SourcePullingAndDeployment or 1001=None.

Get Scan Settings by Project Id - GET /sast/scanSettings/{projectId}

Get scan settings by project Id. In order to retrieve the project Id (id) you should first retrieve the details of all visible projects. To retrieve the details of all visible projects use GET /projects.

Usage

1. GET /projects and get details of all visible projects
2. GET /sast/scanSettings/{projectId} and get scan settings

URL

/cxrestapi/sast/scanSettings/{projectId}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

projectId=[integer] – Unique Id of the project

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Sample Response


```
{
   "project": {
     "id": 1,
     "link": {
       "rel": "project",
       "uri": "/projects/1"
     }
   },
   "preset": {
     "id": 36,
     "link": {
       "rel": "preset",
       "uri": "/sast/presets/36"
     }
   },


  "engineConfiguration": {
     "id": 1,
     "link": {
       "rel": "engineConfiguration",
       "uri": "/sast/engineConfiguration/1"
     }
   }
 }
```


Notes

Retrieves scan settings by project Id. If the request fails, it returns an error response.

Get Scan Settings by Project Id - GET /sast/scanSettings/{projectId}

Get scan settings by project Id. In order to retrieve the project Id (id) you should first retrieve the details of all visible projects. To retrieve the details of all visible projects use GET /projects.

Usage

1. GET /projects and get details of all visible projects
2. GET /sast/scanSettings/{projectId} and get scan settings

URL

http://localhost/cxrestapi/sast/scanSettings/{projectId}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0/1.1

Parameters

Required:

projectId=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/sast/scanSettings/1'

Sample Response


```
{
  "project": {
    "id": 13,
    "link": {
      "rel": "project",
      "uri": "/projects/13"
    }
  },
  "preset": {
    "id": 36,
    "link": {
      "rel": "preset",
      "uri": "/sast/presets/36"
    }
  },
  "engineConfiguration": {
    "id": 1,
    "link": {
      "rel": "engineConfiguration",
      "uri": "/sast/engineConfigurations/1"
    }
  },
  "postScanAction": null,
  "emailNotifications": {
    "failedScan": [],
    "beforeScan": [],
    "afterScan": []
  }
}
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

To retrieve the details of all visible projects use GET /projects. Retrieves scan settings by project Id. If the request fails, it returns an error response. Note that to use this API the user needs to be Scanner at minimum level of permission.

Define SAST Scan Settings - POST /sast/scanSettings

Define the SAST scan settings according to a project (preset and engine configuration).

Usage

1. GET /projects and get details of all visible projects
2. GET /sast/presets and get details of all presets
3. GET /sast/engineConfigurations and get engine configurations list
4. POST /sast/scanSettings and define the SAST scan preset and configuration settings

URL

http://localhost/cxrestapi/sast/scanSettings

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0/1.1
cxOrigin: {request_origin}

Parameters

Required:

scanSettings=[body] – Scan settings:
projectId=[integer] – Unique Id of the project
presetId=[integer] – Unique Id of the preset
engineConfigurationId=[integer] – Unique Id of the engine configuration

Optional:

postScanActionId=[integer] – Unique Id of the post scan action
emailNotifications=[body] – Email notification details:
beforescan=[string] – Specifies the email to send the pre-scan message
failedScans=[string] – Specifies the email to send the scan failure message
afterScans=[string] – Specifies the email to send the post-scan message

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "projectId": 1, \
   "presetId": 1, \
   "engineConfigurationId": 1, \
   "postScanActionId": 1, \
   "emailNotifications": { \
     "failedScan": [ \
       "admin@cx.com" \
     ], \
     "beforeScan": [ \
       "admin@cx.com" \
     ], \      "afterScan": [ \
|       "admin@cx.com" \
     ] \
   } \
 }' 'http://localhost/cxrestapi/sast/scanSettings'
```

Sample Response

```
{
   "id": 2,
   "link": {
     "rel": "self",
     "uri": "/sast/scanSettings/2"
   }
 }
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Defines the SAST scan settings according to a project. If the request fails, it returns an error response. Defines project scan settings regardless of whether the scan has been created.

Update SAST Scan Settings - PUT /sast/scanSettings

Update the SAST scan settings for a project (preset, engine configuration, custom actions and email notifications).

Usage

1. GET /projects and get details of all visible projects
2. GET /sast/presets and get details of all presets
3. GET /sast/engineConfigurations and get engine configurations list
4. PUT /sast/scanSettings and update the SAST scan settings for a project

URL

http://localhost/cxrestapi/sast/scanSettings

Method

PUT

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.1
cxOrigin: {request_origin}

Parameters

Required:

scanSettings=[body] – Scan settings:
projectId=[integer] – Unique Id of the project
presetId=[integer] – Unique Id of the preset
engineConfigurationId=[integer] – Unique Id of the engine configuration

Optional (if already defined, see Notes section):

postScanActionId=[integer] – Unique Id of the post scan action
emailNotifications=[body] – Email notification details:
beforescan=[string] – Specifies the email to send the pre-scan notification
failedScan=[string] – Specifies the email to send the scan failure notification
afterScan=[string] – Specifies the email to send the post-scan notification

Curl Sample:

```
curl -X PUT --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "projectId": 1, \
   "presetId": 1, \
   "engineConfigurationId": 1, \
   "postScanActionId": 1, \
   "emailNotifications": { \
     "failedScan": [ \
       "admin@cx.com" \
     ], \
     "beforeScan": [ \
       "admin@cx.com" \
     ], \
     "afterScan": [ \
       "admin@cx.com" \
     ] \
   } \
 }' 'http://localhost/cxrestapi/sast/scanSettings'
```

Sample Response

```
{
   "id": 2,
   "link": {
     "rel": "self",
     "uri": "/sast/scanSettings/2"
   }
 }
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Updates the SAST scan settings for a project. While certain parameters are optional, if a project already has these details available (i.e. post scan action and email notifications), not providing them when running PUT /sast/scanSettings, will automatically delete them. If the request fails, it returns an error response.

Define SAST Scan Scheduling Settings - PUT /sast/project/{projectId}/scheduling

Define SAST scan scheduling settings for a project.

Usage

1. GET /projects and get details of all visible projects
2. PUT /sast/project/{projectId}/scheduling and update the SAST scan scheduling settings for a project

URL

http://localhost/cxrestapi/sast/project/{projectId}/scheduling

Method

PUT

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

projectId=[integer] – Unique Id of the project

Optional:

scanScheduling=[body] – Scan scheduling details:
scheduleType=[string] – Specifies the schedule type (none or weekly)
scheduledDays=[string] – Specifies the day(s) to perform the scan (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday). Comma separate days can also be defined scheduleTime=[string] – Specifies the time(s) to perform the scan (hh:mm)

Curl Sample:

```
curl -X PUT --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "scheduleType": "weekly", \
   "scheduledDays": [ \
     "Monday", "Thursday" \
   ], \
   "scheduleTime": "23:00" \
 }' 'http://localhost/cxrestapi/sast/project/1/scheduling'
```

Sample Response

no content

Success Response

Code: 204 no content

Error Response

Code: 400 Bad Request

Notes

Defines the SAST scan scheduling settings for a project. If the request fails, it returns an error response.

CxSAST (REST) API - SAST Scan Results

This section includes CxREST APIs for working with scan result tasks.

Assign Ticket to Scan Results - POST /sast/results/tickets

Assign ticket to scan results according to scan results and ticket/issue Id. Ticket/issue Id is usually provided by the ticketing system (e.g. Jira).

Usage

1. GET /sast/scans/{id} and get details of a specific scan
2. POST /sast/results/tickets and assign ticket to scan results

URL

http://localhost/cxrestapi/sast/results/tickets

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0

Parameters

Required:

resultsTicket=[body] – Ticket/results details:
resultsId=[string] – Unique Id of the result. ResultsId is made up of the Scan Id and Path Id (example: ScanId = 1000025, PathId = 12, therefore ResultsId = 1000025-12)
ticketId=[string] – Unique Id of the ticket/issue (see Notes).

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "resultsId": [ \
     "1000025-12" \
   ], \
   "ticketId": "1023" \
 }' 'http://localhost/cxrestapi/sast/results/tickets'
```

Sample Response

no content

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

It is possible to assign a number of results (resultsId) to the same ticket (ticketId). Result Id is made up of the Scan Id and Path Id. Example: Scan Id = 1000025, Path Id = 12, therefore the Result Id = 1000025-12. Please refer to the instruction about how to get JIRA ticket/issue Id.

Publish Last Scan Results to Management and Orchestration by Project Id – POST /sast/projects/{id}/publisher/policyFindings

Publish last scan results to Management and Orchestration for a specific project (only for policy management evaluation in v8.9.0).

Usage

1. GET /projects and get details of all projects
2. POST /sast/projects/{id}/publisher/policyFindings and publish last scan results to Management and Orchestration for a specific project

URL

http://localhost/cxrestapi/sast/projects/{id}/publisher/policyFindings

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X POST --header 'Content-Type: application/json;v=1.0' --header 'Accept: application/json' --header 'Authorization: Bearer <access token> 'http://localhost/cxrestapi/sast/projects/1/publisher/policyFindings'

Sample Response

```
{
  "id": 36,
  "link": {
    "rel": "status",
    "uri": "/sast/projects/36/Publisher/policyFindings/status"
 }
```

Success Response

Code: 201 Created

Error Response

Code: 400 Bad Request

Notes

Publishes the last scan results to Management and Orchestration for a specific project. If the request fails, it returns an error response. Management and Orchestration must be installed in order to run POST /sast/projects/{id}/publisher/policyFindings.

Get the Publish Last Scan Results to Management and Orchestration Status – GET /sast/projects/{id}/publisher/policyFindings/status

Get the status of publish last scan results to Management and Orchestration layer for a specific project.

Usage

1. GET /projects and get details of all projects
2. POST /sast/projects/{id}/publisher/policyFindings and publish last scan results to Management and Orchestration for a specific project.
3. GET /sast/projects/{id}/publisher/policyFindings/status and get the status of publish last scan results to Management and Orchestration for a specific project.

URL

http://localhost/cxrestapi/cxrestapi/sast/projects/{id}/publisher/policyFindings/status

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the project

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/sast/projects/1/publisher/policyFindings/status'

Sample Response


```
{
  "project": {
    "id": 36,
    "link": {
      "rel": "self",
      "uri": "/projects/36"
    }
  },
  "scan": {
    "id": 1000007,
    "link": {
      "rel": "self",
      "uri": "/sast/scans/1000007"
    }
  },
  "status": "Finished",
  "lastSync": "2018-09-12T08:08:20.3393614Z"
}
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves the 'publish last scan results to Management and Orchestration layer' status for a specific project. If the request fails, it returns an error response. Management and Orchestration must be installed in order to run GET /sast/projects/{id}/publisher/policyFindings/status.

Get Short Vulnerability Description for a Scan Result – GET /sast/scans/{id}/results/{pathId}/shortDescription

Get the short version of a vulnerability description for a specific scan result.

Usage

1. GET /reports/sastScans/{id}/status and get the status of the specific report
2. GET /reports/sastScans/{id} and get the report when ready
3. GET /sast/scans/{id}/results/{pathId}/shortDescription and get the short version of the vulnerability description for a specific scan result

URL

http://localhost/cxrestapi/sast/scans/{id}/results/{pathId}/shortDescription

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the scan
pathId=[integer] – Unique Id of the result path

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'CXCSRFToken: 27e4968009e140888c017a4bf639d80e' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/scans/1000002/results/1/shortDescription'

Sample Response


```
{
  "shortDescription": "The application's main method receives an
dynamically executes user-controlled code using line 9 of
\\Code_Injection\\1\\code_injection.java. This could enable an
attacker to inject and run arbitrary code. The attacker can inject the
executed code via user input, which is retrieved by the application in
the main method, at line 4 of
\\Code_Injection\\1\\code_injection.java.\n"
}
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

Retrieves the short version of the vulnerability description for a specific scan result. If the request fails, it returns an error response.

CxSAST (REST) API - SAST Scan Reports

This section includes CxREST APIs for working with scan report tasks.

Get Report(s) by Id - GET /reports/sastScan/{id} (v8.6.0 and up)

Get the specified report once generated. To first generate a report use POST/reports/sastScan.

Usage

1. POST /reports/sastScan and register a new report
2. GET /reports/sastScan/{id}/status and get the status of the specific report
3. GET /reports/sastScan/{id} and get the report when ready

URL

http://localhost/cxrestapi/reports/sastScan/{id}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

Response Content Type=[boolean] – application/rtf, application/xml, application/pdf or application/csv (body of the response is the report content).

id=[string] – Unique Id of the report

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/reports/sastScan/1000057'

Sample Response (application/xml):

```
<?xml version="1.0" encoding="utf-8"?>
 <CxXMLResults InitiatorName="admin" Owner="admin" ScanId="1000005"
ProjectId="2" ProjectName="Project 1"
TeamFullPathOnReportDate="CxServer" DeepLink="http://WIN2K12-
TEMP/CxWebClient/ViewerMain.aspx?scanid=1000005&amp;projectid=2"
ScanStart="Sunday, December 3, 2017 4:50:34 PM" Preset="Checkmarx
Default" ScanTime="00h:03m:18s" LinesOfCodeScanned="6838"
FilesScanned="34" ReportCreationTime="Sunday, December 3, 2017 6:13:45
PM" Team="CxServer" CheckmarxVersion="8.6.0" ScanComments=""
ScanType="Incremental" SourceOrigin="LocalPath" Visibility="Public">
     <Query id="430" categories="PCI DSS v3.2;PCI DSS (3.2) - 6.5.1 -
Injection flaws - particularly SQL injection,OWASP Top 10 2013;A1-
Injection,FISMA 2014;System And Information Integrity,NIST SP 800-
53;SI-10 Information Input Validation (P1),OWASP Top 10 2017;A1-
Injection" cweId="89" name="SQL_Injection" group="CSharp_High_Risk"
Severity="High" Language="CSharp" LanguageHash="1363215419077432"
LanguageChangeDate="2017-12-03T00:00:00.0000000" SeverityIndex="3"
QueryPath="CSharp\Cx\CSharp High Risk\SQL Injection Version:0"
QueryVersionCode="430">
         <Result NodeId="10000050002" FileName="bookstore/Login.cs"
Status="Recurrent" Line="179" Column="103" FalsePositive="False"
everity="High" AssignToUser="" state="0" Remark=""
DeepLink="http://WIN2K12-
TEMP/CxWebClient/ViewerMain.aspx?scanid=1000005&amp;projectid=2&amp;pa
thid=2" SeverityIndex="3">
             <Path ResultId="1000005" PathId="2"
SimilarityId="1765812516">
                 <PathNode>
                     <FileName>bookstore/Login.cs</FileName>
                     <Line>179</Line>
                     <Column>103</Column>
                     <NodeId>1</NodeId>
                     <Name>Text</Name>
                     <Type></Type>
```

```
                    <Length>4</Length>
                    <Snippet>
                        <Line>
                            <Number>179</Number>
                            <Code>  int iPassed =
Convert.ToInt32(Utility.Dlookup("members", "count(*)", "member_login
='" + Login_name.Text + "' and member_password='" +
CCUtility.Quote(Login_password.Text) + "'"));</Code>
                        </Line>
                    </Snippet>
                </PathNode>
                <PathNode>
                    <FileName>bookstore/App_Code.1/CCUtility.cs</File
Name>
                    <Line>168</Line>
                    <Column>59</Column>
                    <NodeId>2</NodeId>
                    <Name>sWhere</Name>
                    <Type></Type>
                    <Length>6</Length>
                    <Snippet>
                        <Line>
                            <Number>168</Number>
                            <Code>  public string Dlookup(string
table, string field, string sWhere)</Code>
                        </Line>
                    </Snippet>
                </PathNode>
                <PathNode>
                    <FileName>bookstore/App_Code.1/CCUtility.cs</File
Name>
                    ……................
```

Success Response

Code: 200 OK

Error Response

Code: 204 No Content
Code: 404 Not Found

Notes

Retrieves the specified report once generated. If the request fails, it returns an error response. The report Id (id) and the report format (application/xml) must be the same as defined when the report is generated; if not an error will occur. Report generated in application/pdf format may not display correctly when using Swagger.

Get Report Status by Id - GET /reports/sastScan/{id}/status

Get the status of a generated report. To retrieve the specified report once the status is known you can use GET /reports/sastScan/{id}. To first generate a report (id) you can use POST /reports/sastScan.

Usage

1. GET /reports/sastScan/{id}/status and get the status of the specific report
2. GET /reports/sastScan/{id} and get the report when ready

URL

http://localhost/cxrestapi/reports/sastScan/{id}/status

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[string] – Unique Id of the report

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/reports/sastScan/1000057/status'

Sample Response


```
{
  "link": {
    "rel": "content",
    "uri": "/reports/sastScan/51"
  },
  "contentType": "application/xml",
  "status": {
    "id": 2,
    "value": "Created"
  }
}
```

Success Response

Code: 200 OK

Error Response

Code: 404 Not Found

Notes

Retrieves the status of a generated report. If the request fails, it returns an error response. Possible statuses are: 0=Deleted, 1=In Process, 2=Created or 3=Failed.

Register Scan Report - POST /reports/sastScan (v8.6.0 and up)

Generate a new CxSAST scan report. Once registered you can use GET /reports/sastScan/{id}/status to get the status of the scan report and then use GET /reports/sastScan/{id} to retrieve the generated report.

Usage

1. POST /reports/sastScan and register a new report
2. GET /reports/sastScan/{id}/status and get the status of the specific report
3. GET /reports/sastScan/{id} and get the report when ready

URL

http://localhost/cxrestapi/reports/sastScan

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

reportRequest=[body] – Report request:
reportType=[string] – Report type options are: PDF, RTF, CSV or XML
scanId=[integer] – Unique Id of the scan

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "reportType": "XML", \
   "scanId": 1000062 \
 }' 'http://localhost/cxrestapi/reports/sastScan'
```

Sample Response

```
{
   "reportId": 6,
   "links": {
     "report": {
       "rel": "content",
       "uri": "/reports/sastScan/6"
     },
     "status": {
       "rel": "status",
       "uri": "/reports/sastScan/6/status"
     }
   }
 }
```

Success Response

Code: 202 Accepted

Error Response

Code: 400 Bad Request

Notes

Generates a new CxSAST scan report. If the request fails, it returns an error response. The report Id (reportId) and the report format (reportType) must be the same as defined when the report is retrieved (GET /reports/sastScan/{id}); if not an error will occur. Report generated in application/pdf format may not display correctly when using Swagger.

CxSAST (REST) API - Engines

This section includes CxREST APIs for working with engine management tasks.

Get All Engine Server Details - GET /sast/engineServers

Get details of all engine servers. The retrieved engine server id (id) can be used to get details of a specific engine server using GET /sast/engineServers/{id}.

Usage

1. GET /sast/engineServers and get details of all engine servers
2. GET /sast/engineServers/{id} and get details of a specific engine server

URL

http://localhost/cxrestapi/sast/engineServers

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

None

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/engineServers'

Sample Response


```
[
  {
    "id": 1,
    "name": "Localhost",
    "uri":
"http://Localhost/CxSourceAnalyzerEngineWCF/CxEngineWebServices.svc",
    "minLoc": 0,
    "maxLoc": 999999999,
    "maxScans": 3,
    "cxVersion": "8.6.0.1947",
    "status": {
      "id": 4,
      "value": "Idle"
    },
    "link": {
      "rel": "self",
      "uri": "/sast/engineServers/1"
    }
  }
]
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves details of all engine servers. If the request fails, it returns an error response. Must be a valid engine server in order to get details. Retrieves all engine details regardless of status. Possible statuses are: 0=Offline, 1=Blocked, 2=Scanning and Blocked, 3=Scanning or 4=Idle.

Register Engine - POST /sast/engineServers

Register a new engine server.

Usage

1. POST /sast/engineServers and register a new engine
2. GET /sast/engineServers/{id} and get details of a specific engine server

URL

http://localhost/cxrestapi/sast/engineServers

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=0.1/1.0
cxOrigin: {request_origin}

URL Parameters

Required:

engineServer=[body] – Engine server details
name=[string] – Name of the engine server
uri=[string] – Specifies the url of the engine server
minLoc=[integer] – Specifies the minimum number of lines of code to scan
maxLoc=[integer] – Specifies the maximum number of lines of code to scan
isBlocked=[Boolean] – Specifies whether or not the engine will be able to receive scan requests
maxScans=[integer] – Specifies the maximum number of concurrent scan to perform

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "name": "Localhost", \
   "uri":
"http://Localhost/CxSourceAnalyzerEngineWCF/CxEngineWebServices.svc",
\
   "minLoc": 0, \
   "maxLoc": 999999999, \
   "isBlocked": true \
 }' 'http://localhost/cxrestapi/sast/engineServers'
```

Sample Response

```
{
   "id": 2,
   "link": {
     "rel": "self",
     "uri": "/sast/engineServers/2"
   }

}
```

Success Response

Code: 201 Created

Error Response

Code: 400 Bad Request

Notes

Registers a new engine server. If the request fails, it returns an error response.

Unregister Engine by Engine Id - DELETE /sast/engineServers/{id}

Unregister an existing engine server. In order to unregister an engine server, you should first retrieve the engine server Id (id). To retrieve the engine server Id (id) use GET /sast/engineServers.

Usage

1. GET /sast/engineServers and get details of all engine servers
2. DELETE /sast/engineServers/{id} and unregister the engine server

URL

http://localhost/cxrestapi/sast/engineServers/{id}

Method

DELETE

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the engine server

Curl Sample:

curl -X DELETE --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/engineServers/2'

Sample Response

No content

Success Response

Code: 204 Deleted

Error Response

Code: 400 Bad Request
Code: 404 Server Id not Found

Notes

Unregisters an existing engine server. If the request fails, it returns an error response. Must be a valid (registered) engine server in order to un-register.

Get Engine Details - GET /sast/engineServers/{id}

Get details of a specific engine server by Id. In order to get details of a specific engine server you should first retrieve the engine server Id (id). To retrieve the engine server Id (id) use GET /sast/engineServers.

Usage

1. GET /sast/engineServers and get details of all engine servers
2. GET /sast/engineServers/{id} and get details of a specific engine server

URL

http://localhost/cxrestapi/sast/engineServers/{id}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[integer] – Unique Id of the engine server

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/engineServers/2'

Sample Response


```
{
   "id": 1,
   "name": "Localhost",
   "uri":
 "http://Localhost/CxSourceAnalyzerEngineWCF/CxEngineWebServices.svc",
   "minLoc": 0,
   "maxLoc": 999999999,
   "maxScans": 2,
   "cxVersion": "8.6.0.1921",
   "status": {
     "id": 4,
     "value": "Idle"
   },
   "link": {
     "rel": "self",
     "uri": "/sast/engineServers/1"
   }
 }
```


Success Response

Code: 200 OK

Error Response

Code: 404 Not Found

Notes

Retrieves details of a specific engine server by Id. If the request fails, it returns an error response. Must be a valid (registered) engine server in order to get details. Possible statuses are: 0=Offline, 1=Blocked, 2=Scanning and Blocked, 3=Scanning or 4=Idle.

Update Engine Server - PUT /sast/engineServers/{id}

Update an existing engine server's configuration and enables to change certain parameters. In order to update an engine server you should first retrieve the engine server details. To retrieve the engine server details use GET /sast/engineServers/{id}.

Usage

1. GET /sast/engineServers/{id} and get details of a specific engine server
2. PUT /sast/engineServers/{id} and update the engine server configuration

URL

http://localhost/cxrestapi/sast/engineServers/{id}

Method

PUT

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=0.1/1.0
cxOrigin: {request_origin}

Parameters

Required:

id=[integer] – Unique Id of the engine server

engineServer=[body] – Engine server details:
name=[string] – Name of the engine server
uri=[string] – Specifies the url of the engine server
minLoc=[integer] – Specifies the minimum number of lines of code to scan
maxLoc=[integer] – Specifies the maximum number of lines of code to scan
isBlocked=[Boolean] – Specifies whether or not the engine will be able to receive scan requests
maxScans=[integer] – Specifies the maximum number of concurrent scans to perform

Curl Sample:

```
curl -X PUT --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "name": "Localhost", \
   "uri": "http://Localhost/CxSourceAnalyzerEngineWCF/CxEngineWebServices.svc", \
   "minLoc": 0, \
   "maxLoc": 999999999, \
   "isBlocked": true \
 }' 'http://localhost/cxrestapi/sast/engineServers/2'
```

Sample Response

```
{
  "id": 2,
  "link": {
    "rel": "self",
    "uri": "/sast/engineServers/2"
  }
}
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request
Code: 404 Not Found

Notes

- Updates an existing engine server's configuration. If the request fails, it returns an error response.
- Block/unblock an engine – An engine can be set as "Blocked" during or after creation by setting the "isBlocked" boolean parameter to "True". A blocked engine will not be able to receive additional scans requests. If the engine is currently running a scan, the running scan will continue until completion.
- Blocking an engine provides an ability to remove an engine once the scan is complete (either temporarily or permanently) without any new scans being appointed to that engine.
- A blocked engine can be unblocked at any given time by setting the "isBlocked" boolean parameter to "False". Once unblocked, the engine will start receiving scan requests.

Get All Engine Configurations - GET /sast/engineConfigurations

Get the engine servers configuration list. You can use the retrieved configuration Id (id) to get a specific engine server's configuration. To get a specific engine server's configuration use GET /sast/engineConfigurations/{id}.

Usage

1. GET /sast/engineConfigurations and get engine server configuration list
2. GET /sast/engineConfigurations/{id} and get specific engine server configuration

URL

http://localhost/cxrestapi/sast/engineConfigurations

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

None

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/engineConfigurations'

Sample Response


```
[
    {
      "id": 1,
      "name": "Default Configuration"
    },
    {
      "id": 2,
      "name": "Japanese (Shift-JIS)"
    },
    {
      "id": 3,
      "name": "Korean"
    },
    {
      "id": 5,
      "name": "Multi-language Scan"
    }
 ]
```


Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Retrieves engine server configuration list. If the request fails, it returns an error response.

Get Engine Configuration by Id - GET /sast/engineConfigurations/{id}

Get a specific engine configuration by configuration Id. In order to retrieve the configuration Id (id) of a specific engine you should first retrieve the engine configuration list. To retrieve the engine configuration list use GET /sast/engineConfigurations.

Usage

1. GET /sast/engineConfigurations and get engine configuration list
2. GET /sast/engineConfigurations/{id} and get specific engine configuration

URL

http://localhost/cxrestapi/sast/engineConfigurations/{id}

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.0

Parameters

Required:

id=[string] – Unique Id of the engine configuration

Curl Sample:

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>' 'http://localhost/cxrestapi/sast/engineConfigurations'

Sample Response

```
{
   "id": 1,
   "name": "Default Configuration"
}
```

Success Response

Code: 200 OK

Error Response

Code: 400 Not Found

Notes

Retrieves a specific engine configuration by configuration Id. If the request fails, it returns an error response. Possible configurations are: 1=Default, 2=Japanese, 3=Korean or 5=Multi-language Scan.

CxSAST (REST) API - Queries

This section includes CxREST APIs for working with project tasks.

Queries - GET /Queries/{queryid}/CxDescription (8.6.0 and up)

Get long description for the query id.

Usage

GET /Queries/{queryid}/CxDescription

URL

/cxrestapi/help/Queries/{queryid}/CxDescription

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>

Parameters

id=[Long] – Unique Id of the query

Curl Sample

curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer <access token>
'http://localhost/cxrestapi/help/Queries/{queryid}/CxDescription'

Success Response

Code: 200 OK

Error Response

Code: 404 Not Found

Notes

Retrieves details of all teams. If the request fails, it returns an error response.

CxSAST (REST) API - Data Retention

This section includes CxREST APIs for working with data retention tasks.

Stop Data Retention - POST /sast/dataRetention/stop

Stop the data retention (global)

Usage

POST /sast/dataRetention/stop

URL

http://localhost/cxrestapi/sast/dataRetention/stop

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.1
cxOrigin: {request_origin}

Parameters

Required:

None

Curl Sample:

curl -X POST --header 'Content-Type: application/json;v=1.0' --header 'Accept: application/json' --header 'Authorization: Bearer <access token> 'http://localhost/cxrestapi/sast/dataRetention/stop'

Sample Response

no content

Success Response

Code: 202 Accepted

Error Response

Code: 400 Bad Request

Notes

Stops the data retention globally. If the request fails, it returns an error response.

Define Data Retention Date Range - POST /sast/dataRetention/byDateRange

Define the global setting for data retention by date range

Usage

POST /sast/dataRetention/byDateRange

URL

http://localhost/cxrestapi/sast/sast/dataRetention/byDateRange

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.1
cxOrigin: {request_origin}

Parameters

Required:

dataRetentionByDates=[body] – Data Retention by Date Range details:
startDate=[string] – Data retention start date
endDate=[string] – Data retention end date
durationLimitInHours=[integer] – Duration limit (in hours)

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "startDate": "2019-06-17", \
   "endDate": "2019-06-18", \
   "durationLimitInHours": 1 \
 }' 'http://localhost/cxrestapi/sast/dataRetention/byDateRange'
```

Sample Response

```
{
  "id": 3,
  "link": {
    "rel": "status",
    "uri": "sast/dataRetention/3/status"
  }
}
```

Success Response

Code: 202 Accepted

Error Response

Code: 400 Bad Request

Notes

Defines the global setting for data retention by date range. For data retention, the time is included in all date requests/results, although time is ignored in principle (v8.8.0 only). If the request fails, it returns an error response.

Define Data Retention by Number of Scans - POST /sast/dataRetention/byNumberOfScans

Define the global setting for the data retention by number of scans.

Usage

POST /sast/dataRetention/byNumberOfScans

URL

http://localhost/cxrestapi/sast/dataRetention/byNumberOfScans

Method

POST

Media Type (Header)

Authorization: Bearer <access token value>
Content-Type: application/json;v=1.1
cxOrigin: {request_origin}

Parameters

Required:

dataRetentionByNumberOfScans=[body] – Data Retention Number of Scans details:
numOfSuccessfulScansToPreserve=[integer] – Number of successful scans to keep
durationLimitInHours=[integer] – Duration limit (in hours)

Curl Sample:

```
curl -X POST --header 'Content-Type: application/json;v=1.0' --header
'Accept: application/json' --header 'Authorization: Bearer <access
token> -d '{ \
   "numOfSuccessfulScansToPreserve": 3, \
   "durationLimitInHours": 1 \
 }' 'http://localhost/cxrestapi/sast/dataRetention/byNumberOfScans'
```

Sample Response

```
{
  "id": 3,
  "link": {
    "rel": "status",
    "uri": "sast/dataRetention/3/status"
  }
}
```

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Notes

Defines the global setting for data retention by date range. For data retention, the time is included in all date requests/results, although time is ignored in principle (v8.8.0 only). If the request fails, it returns an error response.

Get Data Retention Request Status - GET /sast/dataRetention/{requestId}/status

Get status details of a specific data retention request.

Usage

1. POST sast\dataRetention\byNumberOfScans and define the global setting for data retention by date range
2. GET /sast/dataRetention/{requestId}/status and get details of the data retention request.

URL

cxrestapi/sast/dataRetention/{requestId}/status

Method

GET

Media Type (Header)

Authorization: Bearer <access token value>
Accept: application/json;v=1.1

Parameters

Required:

requestId=[integer] – Unique Id of the data retention request.

Success Response

Code: 200 OK

Error Response

Code: 400 Bad Request

Sample Response

```
{
  "id": 1,
  "stage": {
    "id": 8,
    "value": "Finished"
  },
  "link": {
    "rel": "self",
    "uri": "/sast/dataRetention/1/status"
  }
}
```

Notes

Retrieves status details of a specific data retention request. If the request fails, it returns an error response. The requestId is returned by POST sast\dataRetention\byNumberOfScans or POST /sast/dataRetention/byDateRange.

CxSAST (REST) API - Open Source Analysis (CxOSA)

For more information about CxREST APIs for working with open source analysis tasks and Open Source Analysis (CxOSA) in general, see CxOSA (REST) API - Open Source Analysis in the Checkmarx CxOSA Documentation.

CxSAST (REST) API - Swagger Examples

Checkmarx's latest REST API's (CxREST API) are documented using Swagger. Swagger is a powerful open source framework backed by a large ecosystem of tools that is used to design, build, document, and consume RESTful APIs (see http://swagger.io/ for more information).

This section includes Swagger examples for working with CxREST APIs.

To access a live Swagger environment navigate to: http://<ServerName>/cxrestapi/help/swagger/ui/index (e.g. http://localhost/cxrestapi/help/swagger/ui/index)

Authentication is required for exploring the CxREST API through Swagger. Click the authentication indicator and you will be redirected to the relevant Login page.

CxSAST (REST) API - Swagger Examples (v9.0)

Please note that in this version swagger is only used to display api examples and is in no shape or form an interactive tool for testing api's.

To access a live Swagger environment navigate to: http://<ServerName>/cxrestapi/help/swagger/ui/index (e.g. http://localhost/cxrestapi/help/swagger/ui/index)

Authentication is required for exploring the CxREST API through Swagger. Click the authentication indicator and you will be redirected to the relevant Login page.

CxSAST (SOAP) API

The following section includes information about developing client implementations using the CxSAST SOAP API.

Before creating CxSAST API clients, it is recommended to become familiar with CxSAST concepts.

SOAP API Overview

Introduction

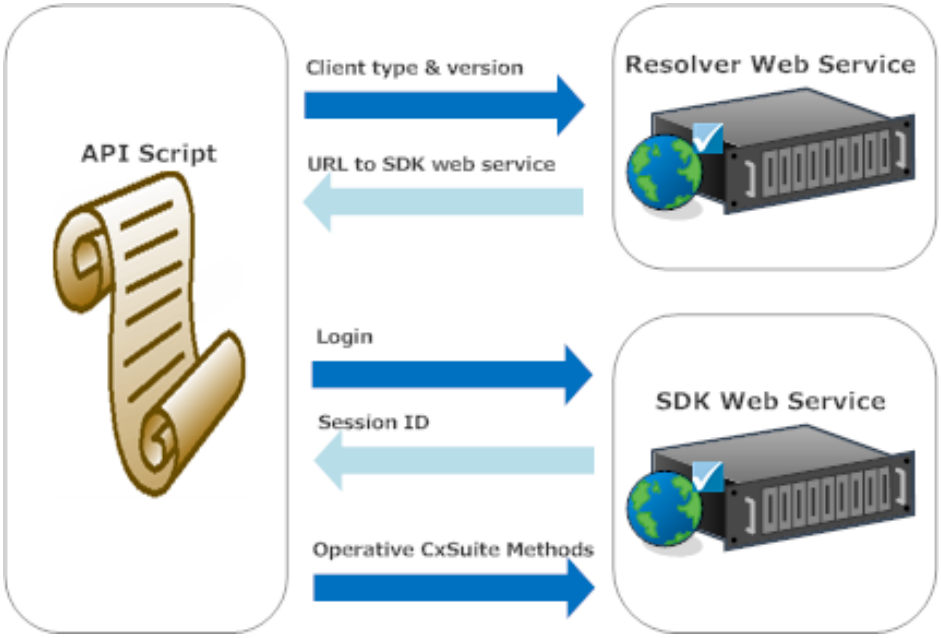
The CxSAST API provides developers with the ability to create client scripts for working with CxSAST projects, code scans, and result reports.

API Architecture and Authentication

CxSAST provides a separate web service for each version of each CxSAST client (such as the web console, the IDE plugins, and this API). An additional Resolver web service performs the sole function of providing clients with the appropriate web service URL for the client. So, any API client script needs to access two CxSAST web services:

1. First, the client should access the Resolver web service (class: CxWSResolver), and use its single method to submit the client type (a fixed value representing that it is an API client rather than one of the other CxSAST clients) and the API version for which the script was written. The web service then returns the SDK web service URL for the appropriate API.
2. The client can then access the appropriate SDK web service (class: CxSDKWebService), as obtained from the Resolver. All subsequent CxSAST API methods will be to this SDK web service.

API sessions to the SDK web service must be initiated by using the login method, which returns a Session ID. This Session ID must be submitted in all subsequent methods.



API Method Responses and Error Handling

The response objects returned by API methods all include the following fields:

- **.IsSuccesfull** (boolean): Upon any error(s) resulting from the method, this field's value is set to **false**.
- **.ErrorMessage** (string): Upon any error(s) resulting from the method, the error is specified in this field.
- Additional method-specific fields are described with the methods, where relevant.

Mapping SOAP to REST

This section is designed to be used as a basic summary of SOAP to REST API mapping. SOAP APIs are grouped according to their product area and each API has a direct link to the relevant API documentation. Mapping for each SOAP API and it's related REST API is also indicated. Additional information is also provided. It is also highly important to see, New REST APIs and Authentication Methods – Upgrade Implications.

Group	SOAP API	REST API	Additional Information
Login	Login	POST /auth/login	SOAP cookie-based login replaced with a REST Token: Token-based Authentication / Login using OAuth 2.0.
	Logout LoginWithToken SsoLogin	See above	See above
Projects	IsValidProjectName		Replaced with GET /projects.
	GetProjectConfiguration	GET /projects	Get details of all projects.
		GET /projects/{id}	Get details of a specific project.
	GetPresetList	GET /sast/presets	Get details of all presets.
	UpdateProjectConfiguration	PUT /projects/{Id}	Update specific project's details. Parameters include - name, owningTeam and customFields (Id and value).
		GET /customFields	Get details of all custom fields.
		POST /sast/scanSettings	Define specific project's scan settings. Parameters include - presetId, engineConfigurationId, postScanActionId and emailNotifications (beforescan, failedScans, afterScans).

Group	SOAP API	REST API	Additional Information
		PUT /projects/{id}/sourceCode/excludeSettings	Set a specific project's exclude folders/files settings. Parameters include - excludeFoldersPattern and excludeFilesPattern.
		PUT /sast/project/{projectId}/scheduling	Define specific project's scan scheduling settings. Parameters include - scheduleType and scheduleDays.
		POST /projects/{id}/issueTrackingSettings/jira	Set a specific project's Jira issue tracking system settings. Parameters include - issueTrackingSystemId, jiraProjectId, issueType, field Ids and values.
		POST /projects/{id}/dataRetentionSettings	Set a specific project's data retention settings. Parameters include - scansToKeep.
	GetConfigurationSetList	GET /sast/engineConfiguration	Get details of all engine configurations.
	DeleteProjects	DELETE projects/{projectId}	Delete a specific project. Parameters include - deleteRunningScans (true/false).
	BranchProjectById	POST /projects/{id}/branch	Create a specific project's branch. Parameters include - name.
	GetProjectScannedDisplayData	GET /sast/scans?projectId={projectId}	Get all scans for a specific project.
	GetProjectsDisplayData	GET /projects	Gets details of all projects. Returns wide-ranging project information - owning team, latest scan, all project scans, scan settings and custom fields.
		GET /customFields	Get details of all custom fields.
		PUT /projects/{id}	Update an existing project's details. Parameters include – name, owningTeam and customFields (id and value).
Scans	Scan	POST /sast/scanSettings	Define a specific project's scan settings. Parameters include - presetId, engineConfigurationId, postScanActionId and emailNotifications (beforescan, failedScans, afterScans).
		POST /projects/{Id}/sourceCode/attachments	Upload a specific project's zip file (contains the source code for scanning). Parameters include - zippedSource.
		POST /projects/{Id}/sourceCode/remoteSettings/git	Set a specific project's remote source settings for GIT. Parameters include - url, branch and privateKey.

Group	SOAP API	REST API	Additional Information
		GET /projects/{Id}/sourceCode/remoteSettings/git	Get a specific project's remote source settings for GIT.
		POST /projects/{Id}/sourceCode/remoteSettings/git/ssh	Set a specific project's remote source settings for GIT using SSH. Parameters include - url, branch and privateKey.
		POST /projects/{Id}/sourceCode/remoteSettings/svn	Set a specific project's remote source settings for SVN. Parameters include - url, absoluteUrl, port, paths and credentials (username, password and privateKey).
		GET /projects/{Id}/sourceCode/remoteSettings/svn	Get a specific project's remote source settings for SVN.
		POST /projects/{Id}/sourceCode/remoteSettings/svn/ssh	Set a specific project's remote source settings for SVN using SSH. Parameters include - absoluteUrl, port, paths and privateKey.
		POST /projects/{Id}/sourceCode/remoteSettings/tfs	Set a specific project's remote source settings for TFS. Parameters include - credentials (username and password), url, absoluteUrl, port and paths.
		GET /projects/{Id}/sourceCode/remoteSettings/tfs	Get a specific project's remote source settings for TFS.
		POST /projects/{Id}/sourceCode/remoteSettings/perforce	Set a specific project's remote source settings for Perforce. Parameters include - credentials (username and password), url, absoluteUrl, port, paths and browseMode.
		GET /projects/{Id}/sourceCode/remoteSettings/perforce	Get a specific project's remote source settings for Perforce.
		POST /projects/{Id}/sourceCode/remoteSettings/shared	Set a specific project's remote source settings for a shared repository. Parameters include – paths and credentials (username and password).
		GET /projects/{Id}/sourceCode/remoteSettings/shared	Get a specific project's remote source settings for a shared repository.
		POST /projects/{Id}/sourceCode/remoteSettings/custom	Set a specific project's remote source settings for a custom repository (e.g. source pulling). Parameters include – paths and credentials (username and password).
		GET /projects/{Id}/sourceCod	Get a specific project's remote source settings for a custom repository (e.g. source pulling). Parameters include – paths,

Group	SOAP API	REST API	Additional Information
		e/remoteSettings/custom	preScanCommandId and credentials (username and password).
		POST /sast/scans	Create a new scan and assign it to a specific project. Parameters include – isIncremental, isPublic, forceScan and comment.
	DeleteScans	DELETE /sast/scans/{id}	Delete a specific scan.
	CancelScan	PATCH /sast/scansQueue/{id}	Cancel a specific scan while still in the queue. Parameters include - status (cancelled).
	UpdateProjectIncrementalConfiguration	POST /sast/scanSettings	Define a specific project's scan settings. Parameters include - presetId, engineConfigurationId, postScanActionId and emailNotifications (beforescan, failedScans, afterScans).
	UpdateScanComment	PATCH /sast/scans/{id}	Add a comment to a specific scan. Parameters include - comment.
	ScanWithOriginName	POST /sast/scans	Custom name added to CxOrigin in the POST /sast/scans header.
	ScanWithScheduling		Merged with ScanWithSchedulingWithCron.
	ScanWithSchedulingWithCron	PUT /sast/project/{projectId}/scheduling	Define specific project's scan scheduling settings. Parameters include - scheduleType and scheduleDays.
		GET /sast/scans/{id}	Get details of a specific scan. Returns status and stage of the scan.
		GET /sast/scanSettings/{projectId}	Get a specific project's scan settings. Returns preset and engine configuration of the scan.
		GET /sast/scansQueue	Get details of all scans in the scans queue. Returns wide-ranging scan information (e.g. stageDetails, engineId, languages, teamId, loc, origin, queuePosition, isIncremental, isPublic, origin, creation date, etc..).
		GET sast/scans?scanStatus={status}	Get all scans with a specific scan status (Scanning, Finished, Canceled or Failed).
		GET /sast/scans?last={numberOfLastScans}	Get all scans according to number of last scans.
		GET /sast/scans	Get all scans.

Group	SOAP API	REST API	Additional Information
	GetStatusOfSingleScan	GET /sast/scansQueue/{Id}	Get details of a specific scan in the scans queue.
	GetScanSummary	GET /sast/scans	Get all scans. Enhanced API with detailed scan information similar to SOAP.
	GetScansDisplayDataForAllProjects	GET /sast/scans	Get all scans. Enhanced API with detailed scan information similar to SOAP. Get the last scan of a project.
		Get sast/scans/{id}/resultsStatistics	Get statistic results for a specific scan. Returns summary of results (by severity). Result is also available as a link in the GET /sast/scans resource.
		Get /sast/scans?projectId={projectId}&Last={number}	Get the last scan of a specific project.
Scan Reports	CreateScanReport	POST /reports/sastScan	Generate a new scan report.
	GetScanReportStatus	GET /reports/sastScan/{Id}/status	Get the status of a generated report.
	GetScanReport	GET /reports/sastScan/{Id}	Get the specific report once generated.
Managing Users	GetAllUsers	For future release	
	DeleteUser	For future release	
	GetAssociatedGroupsList	GET /auth/teams	Gets details of all teams.
	GetTeamLdapGroupsMapping	For future release	
	SetTeamLdapGroupsMapping	For future release	
Data Retention	ExecuteDataRetention	POST /sast/dataRetention/byDateRange	Define data retention global settings by date range. Parameters include – startDate, endDate and durationLimitInHours.
		POST /sast/dataRetention/byNumberOfScans	Define data retention global settings by number of scans. Parameters include – numOfSuccessfulScansToPreserve and durationLimitInHours.

Group	SOAP API	REST API	Additional Information
	StopDataRetention	POST /sast/dataRetention/stop	Stops global data retention.
		POST /projects/{Id}/dataRetentionSettings	Set specific project's data retention settings. Parameters include – scansToKeep.

Getting the SDK Web Service URL

In order to initiate a session to the SDK web service (which receives operative CxSAST requests), the API client must first retrieve its URL from the Resolver web service. The API client should do the following:

1. Create an instance of the Resolver generated proxy client (class: **CxWSResolver**), and set its URL to:
   `http://<server>/Cxwebinterface/CxWsResolver.asmx`
   where `<server>` is the IP address or resolvable name of the CxSAST server (in a distributed architecture: the CxManager server).
2. Call the Resolver's single available method (GetWebServiceUrl) as below. The returned CxWSResponseDiscovery object's .ServiceURL field will contain the SDK web service URL.

CxWSResolver.GetWebServiceUrl Method

```
public CxWSResponseDiscovery GetWebServiceUrl(
   CxClientType ClientType,
   int APIVersion
);
```

Parameters

- **ClientType**: For API clients, the value is: **SDK**
- **APIVersion**: The current API version is: **1**

Return Value

The SDK web service URL, in **CxWSResponseDiscovery.ServiceURL**

Example

```
internal void Main(string [] args)
{
    CxWSResolverSoapClient cxResolverProxy = new
CxWSResolverSoapClient();
    int APIVersion = 1;
    CxWSResponseDiscovery response=
cxResolverProxy.GetWebServiceUrl(SDK, APIVersion);
    URL = response.ServiceURL;
}
```

Initiating a Session

To be able to submit operative requests, the API client needs to first initiate a session to the SDK web service by creating an instance of the SDK generated proxy client (class: **CxSDKWebService**) with the SDK Web Service URL and calling the Login method as below. The method returns the CxWSResponseLoginData object, which includes the **SessionId** field that needs to be submitted in all subsequent methods.

CxSDKWebService.Login Method

```
public CxWSResponseLoginData Login(
   Credentials applicationCredentials,
   int lcid
);
```

Parameters

- **applicationCredentials**: A **Credentials** object, with fields:
  - **User**: The username for login
  - **Pass**: The password for login
- **lcid**: ID# of the language for web service responses. The current API version supports the following values:
  - **1033**: English
  - **1028**: Chinese Taiwan
  - **1041**: Japanese
  - **2052**: Chinese

Return Value

User data, including the **.SessionId** field

Example

To initiate a session using the CxSAST default credentials (admin@cx, admin), to be receiving English responses:

```
public void LogAdminIn()
{
  CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();
  CxWSResponseLoginData loginResult = cxSDKProxy.Login(new
Credentials() { User = "admin@cx", Pass = "admin" }, 1033);
  sessionID = loginResult.SessionId;
}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

Login	POST /auth/login	SOAP cookie-based login replaced with a REST Token: Token-based Authentication / Login using OAuth 2.0.
Logout LoginWithToken SsoLogin	See above	See above

For more mapping information, refer to Mapping SOAP to REST (v8.8.0 and up). You can also find a summary of our REST APIs here.

Working with CxSAST Projects

This section includes SDK methods for working with CxSAST projects.

The current version of the CxSAST API does not provide a method for creating a project. Instead, first use the scan method with a new project name, which will create the project (and immediately run a scan); then, you can configure the project.

Getting Projects for Display

The API client can get a list of CxSAST projects available to the current user. This is used primarily for display purposes.

CxSDKWebService.GetProjectsDisplayData Method

```
public CxWSResponseProjectsDisplayData GetProjectsDisplayData(
   string sessionID
);
```

Parameters

- **sessionID**: The current Session ID

Return Value

Array of projects. Each project contains data for display.

Example

```
internal void Main(string [] args)

{

   String sessionID = args[0];

   CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



   //request for all the projects related to current user

   CxWSResponseProjectsDisplayData response=
cxSDKProxy.GetProjectsDisplayData(sessionID);



  //set the project data mainly for display use

   ProjectsData = response.projectList;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetProjectsDisplayData	GET /projects	Gets details of all projects. Returns wide-ranging project information - owning team, latest scan, all project scans, scan settings and custom fields.
	GET /customFields	Get details of all custom fields.
	PUT /projects/{id}	Update an existing project's details. Parameters include – name, owningTeam and customFields (id and value).

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Getting Project Details

The API client can get configuration details and user permissions for a known project.

The obtained configuration detail object can be used to change project configuration.

CxSDKWebService.GetProjectConfiguration Method

```
public CxWSResponseProjectConfig GetProjectConfiguration(
   string sessionID,
   long projectID
);
```

Parameters

- **sessionID**: The current Session ID
- **projectID**: Project identifier. Can be obtained from the server response upon running a scan.

Return Value

Project details, including the following fields:

- **.ProjectConfig**: A Project Configuration object
- **.Permission**: User permissions for this project

Example

To get project details for a project with a known ID of **200**:

```
internal void Main(string [] args)
{
   String sessionID = args[0];
   CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();

   //The project unique ID
   long projectID = 200;

   CxWSResponseProjectConfig response=
cxSDKProxy.GetProjectConfiguration(sessionID, projectID);

   //project configuration
   ProjConfig = response.ProjectConfig;

   //user permissions for the project
   Permissions = response.Permission;
}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetProjectConfiguration	GET /projects	Get details of all projects.
	GET /projects/{id}	Get details of a specific project.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Getting Available Query Presets

The API client can get a list of query presets available to the current user.

CxSDKWebService.GetPresetList Method

```
public CxWSResponsePresetList GetPresetList(
   string SessionID
);
```

Parameters

sessionID: The current Session ID

Return Value

Array of query presets.

Example

```
internal void Main(string [] args)

{

    String sessionID = args[0];

    CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();

    CxWSResponsePresetList response =
cxSDKProxy.GetPresetList(sessionID);

    PresetListSucceded = response.IsSuccesfull;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetPresetList	GET /sast/presets	Get details of all presets

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Branching a Project

The API client can create a branch for an existing project.

To create a new project, first run a scan with a new project name, and then branch the existing project as described here.

CxSDKWebService.BranchProjectById Method

```
public CxWSResponseRunID BranchProjectById(
 string sessionId,
 long originProjectId,
 string newBranchProjectName
);
```

Parameters

- **sessionId**: The current Session ID
- **originProjectId**: The original Project ID of the project to be branched. Can be obtained from the server response upon running a scan.
- **newBranchProjectName**: New Project Name for the branch.

Return Value

CxWSResponseRunID, including:

- **.ProjectID**: ID of the newly-created (branched) project. Can be subsequently used to get project details or to configure a project.
- **.RunId**: Unique ID of the scan. Can be subsequently used to get scan status and details.

If there are any errors, the return value parameter "**CxWSResponseRunID.IsSuccesfull**" is set to false and the second parameter "**CxWSResponseRunID.ErrorMessage**" indicates the reason for failure.

Example

To branch an existing project with a known ID of **200**:

Visual Studio Example:

```
internal void Main(string [] args)

{

   String sessionID = args[0];

   CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



   //The project unique ID

   long projectID = 200;



   CxWSResponseRunID response= cxSDKProxy.BranchProjectById(sessionID,
projectID,"NewProjectName");



   //project configuration Id

   Long projectId = response.ProjectID;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

BranchProjectById	POST /projects/{id}/branch	Create a specific project's branch. Parameters include - name.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Configuring a Project

The API client can change the configuration of an existing project. To create a new project, first run a scan with a new project name, and then configure the project as described here.

CxSDKWebService.UpdateProjectIncrementalConfiguration Method

```
public CxWSBasicRepsonse UpdateProjectIncrementalConfiguration(

   string sessionID,

   long projectID,

   ProjectConfiguration projectConfiguration

);
```

Parameters

- **sessionID**: The current Session ID
- **projectID**: The Project ID of the project to be configured. Can be obtained from the server response upon running a scan.
- **projectConfiguration**: An instance of class ProjectConfiguration, to be applied as the new settings for the project. Existing settings can be obtained by getting project details and storing, from the response, **CxWSResponseProjectConfig.ProjectConfig** . The APi client can then modify the stored settings and re-apply the object to the project (see example).

Example

To modify existing settings of a project with a known ID of 200, changing the schedule to Now:

```
internal void Main(string [] args)

{

   String sessionID = args[0];

   CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



   long projectID = 200;



   //get project configuration
```

```
   CxWSResponseProjectConfig ProjConfigResponse =
cxSDKProxy.GetProjectConfiguration(sessionID, projectID);

   ProjectConfiguration ProjConfig = ProjConfigResponse.ProjectConfig;



   //modify schedule settings

   ProjConfig.ScheduleSettings.Schedule = ScheduleType.Now;



   //Apply modified settings to project

   CxWSBasicRepsonse response =
cxSDKProxy.UpdateProjectIncrementalConfiguration(sessionID, projectID,
ProjConfig);

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

UpdateProjectIncrementalConfiguration	POST /sast/scanSettings	Define a specific project's scan settings. Parameters include - presetId, engineConfigurationId, postScanActionId and emailNotifications (beforescan, failedScans, afterScans).

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Project Configuration Members

The API client can retrieve a project configuration object from a given project. A project configuration object needs to be submitted as a method argument when running a scan or when configuring a project.

The project configuration object is of class CliScanArgs / ProjectConfiguration, and it includes:

- **.PrjSettings**: An instance of class **ProjectSettings**, which includes the following fields:
  - **.ProjectName**: Full path on CxSAST server to project (string, beginning with "CxServer\"). You can find this path in the CxSAST web interface. When being submitted as an argument in the Scan method, if the project doesn't already exist, CxSAST will create a new one.
  - **.PresetID**: ID# of query preset to be used. You can find query preset IDs in the CxSAST database, in the dbo.Presets table.
  - **.ScanConfigurationID**: ID# of character encoding used in code project files. You can find encoding IDsin the CxSAST database, in the dbo.configurations table; or, the API client can get available encoding options.
- **.SrcCodeSettings**: An instance of class **SourceCodeSettings**, which includes the following fields:
  - **.SourceOrigin**: One of the following (Source Pulling is not supported for API clients):
    - SourceLocationType.Local
    - SourceLocationType.Shared
    - SourceLocationType.SourceControl
  - **.PackagedCode**: An instance of class LocalCodeContainer, for local scans. Includes the following fields:
    - **.FileName**: Path to ZIP archive of code project (string, for a Local scan).
    - **.ZippedFile**: Byte array containing the contents of the code project. See example.
- **.ScheduleSettings.Schedule**: When to scan. Ignored when being submitted as an argument in the Scan method.
- **.IsPrivateScan**: Whether the scan results should be private, that is, only for the current user (boolean).
- **.IsIncremental**: Whether the scan should be only of new and modified files since the last previous scan (boolean).

Getting Scanned Projects

The API client can get a list of all public projects in the system with a risk level and summary of results by severity (high, medium, low).

CxSDKWebService.GetProjectScannedDisplayData Method

```
public CxWSResponseProjectScannedDisplayData
GetProjectScannedDisplayData(string sessionID);
```

Parameters

sessionID: The current Session ID.

Return Value

CxWSResponseProjectScannedDisplayData, including:

- **.ProjectScannedList**: The list of all public projects in the system with risk level

Example

To get the list of all public projects with risk level:

```
internal void Main(string [] args)

{

  String sessionID = args[0];

  CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();

  CxWSResponseScanStatus response = cxSDKProxy.
GetProjectScannedDisplayData(sessionID);



    //store the Scanned list

    ScannedList

  = response.ProjectScannedList;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetProjectScannedDisplayData	GET /sast/scans?projectId={projectId}	Get all scans for a specific project.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Deleting a Project

The API client can delete an existing project and all related scans. Scans that are currently running are stopped and deleted together with the project. If there's even a single scan that cannot be deleted (due to security reasons) the operation is marked as failed and an error message is returned.

cxSDKProxy.DeleteProjects Method

```
public CxWSBasicRepsonse DeleteProjects(

   string sessionID,

   long projectIDs,

);
```

Parameters

- **sessionID**: The current Session ID
- **projectIDs**: The Project IDs of the project to be deleted. Can be obtained from the server response upon running a scan.

Return Value

If there are any errors, the return value parameter “CxWSBasicRepsonse.IsSuccesfull” is set to false and the second parameter “CxWSBasicRepsonse.ErrorMessage” indicates the reason for failure.

Example

```
To delete existing projects and its related scans:

internal void Main(string [] args)

{

  String sessionID = args[0];

  CxSDKWebService cxSDKProxy = new CxSDKWebService();



  //the projects unique id

  long[] projectsIDs = new long[]{1,2,3};



  //cancel the scan

  CxWSBasicRepsonse response = cxSDKProxy.DeleteProjects(sessionID,
projectsIDs);

 }
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

DeleteProjects	DELETE projects/{projectId}	Delete a specific project. Parameters include - deleteRunningScans (true/false).

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Working with Scans

This section includes SDK methods for working with scans.

Running a Scan

The API client can call an immediate scan. Depending on whether the submitted project name (CliScanArgs.PrjSettings.ProjectName) already exists, the scan is called for the existing CxSAST project or a new project is created.

CxSDKWebService.Scan Method

```
public CxWSResponseRunID Scan(

   string sessionId,

   CliScanArgs args

);
```

Parameters

- **sessionID**: The current Session ID
- **args**: An instace of class **CliScanArgs**, containing project configuration.

When scanning to an existing CxSAST project, get the existing project configuration, modify as needed, and submit the modified project configuration object.

Return Value

CxWSResponseRunID, including:

- **.ProjectID**: ID of the existing or newly-created project. Can be subsequently used to get project details or to configure a project.
- **.RunId**: Unique ID of this scan. Can be subsequently used to get scan status and details.

Example 1

To scan from a local source to a new project:

```
internal void Main(string [] args)

{

   String sessionID = args[0];

   CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



   ProjectSettings projSettings = new ProjectSettings();



   //The project full name

   projSettings.ProjectName = @"CxServer\SP\Company\NewScanProject";



   //Set the query preset to 'default' preset (ID = 7)

   projSettings.PresetID = 7;



   //Set the source files encoding, English = 1

   projSettings.ScanConfigurationID = 1;



   SourceCodeSettings sourceCodeSettings = new SourceCodeSettings();



   //Set the source code location to be local

   sourceCodeSettings.SourceOrigin = SourceLocationType.Local;



   //Set the zipped file and put its contents into byte array
```

```
   sourceCodeSettings.PackagedCode = new LocalCodeContainer();

   sourceCodeSettings.PackagedCode.FileName =
@"C:\Server\Sources.zip";

   sourceCodeSettings.PackagedCode.ZippedFile =
File.ReadAllBytes(sourceCodeSettings.PackagedCode.FileName);




   CliScanArgs scanArgs = new CliScanArgs();

   scanArgs.PrjSettings = projSettings;

   scanArgs.SrcCodeSettings = sourceCodeSettings;


   //The scan is public for all users

   scanArgs.IsPrivateScan = false;


   //Scan all sources, not just changed sources

   scanArgs.IsIncremental = false;


   CxWSResponseRunID response = cxSDKProxy.Scan(sessionID, scanArgs);

   ScanSucceded = response.IsSuccesfull;

   RunID = response.RunId;

   PrjctID = response.ProjectID;

}
```

Example 2

To call an immediate scan to an existing project with a known ID of **200**, maintaining existing project settings:

```
internal void Main(string [] args)
```

```
{

   String sessionID = args[0];

   CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



   //Get existing project settings

   long projectID = 200;

   CxWSResponseProjectConfig ProjectSettingsResponse =
cxSDKProxy.GetProjectConfiguration(sessionID, projectID);

   ProjConfig = ProjectSettingsResponse.ProjectConfig;



   CxWSResponseRunID response = cxSDKProxy.Scan(sessionID,
ProjConfig);

   ScanSucceded = response.IsSuccesfull;

   RunID = response.RunId;

   PrjctID = response.ProjectID;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

Scan	POST /sast/scanSettings	Define a specific project's scan settings. Parameters include - presetId, engineConfigurationId, postScanActionId and emailNotifications (beforescan, failedScans, afterScans).
	POST /projects/{Id}/sourceCode/attachments	Upload a specific project's zip file (contains the source code for scanning). Parameters include - zippedSource.
	POST /projects/{Id}/sourceCode/remoteSettings/git	Set a specific project's remote source settings for GIT. Parameters include - url, branch and privateKey.
	GET /projects/{Id}/sourceCode/remoteSettings/git	Get a specific project's remote source settings for GIT.

	POST /projects/{Id}/sourceCode/remoteSettings/git/ssh	Set a specific project's remote source settings for GIT using SSH. Parameters include - url, branch and privateKey.
	POST /projects/{Id}/sourceCode/remoteSettings/svn	Set a specific project's remote source settings for SVN. Parameters include - url, absoluteUrl, port, paths and credentials (username, password and privateKey).
	GET /projects/{Id}/sourceCode/remoteSettings/svn	Get a specific project's remote source settings for SVN.
	POST /projects/{Id}/sourceCode/remoteSettings/svn/ssh	Set a specific project's remote source settings for SVN using SSH. Parameters include - absoluteUrl, port, paths and privateKey.
	POST /projects/{Id}/sourceCode/remoteSettings/tfs	Set a specific project's remote source settings for TFS. Parameters include - credentials (username and password), url, absoluteUrl, port and paths.
	GET /projects/{Id}/sourceCode/remoteSettings/tfs	Get a specific project's remote source settings for TFS.
	POST /projects/{Id}/sourceCode/remoteSettings/perforce	Set a specific project's remote source settings for Perforce. Parameters include - credentials (username and password), url, absoluteUrl, port, paths and browseMode.
	GET /projects/{Id}/sourceCode/remoteSettings/perforce	Get a specific project's remote source settings for Perforce.
	POST /projects/{Id}/sourceCode/remoteSettings/shared	Set a specific project's remote source settings for a shared repository. Parameters include – paths and credentials (username and password).
	GET /projects/{Id}/sourceCode/remoteSettings/shared	Get a specific project's remote source settings for a shared repository.
	POST /projects/{Id}/sourceCode/remoteSettings/custom	Set a specific project's remote source settings for a custom repository (e.g. source pulling). Parameters include – paths and credentials (username and password).
	GET /projects/{Id}/sourceCode/remoteSettings/custom	Get a specific project's remote source settings for a custom repository (e.g. source pulling). Parameters include – paths, preScanCommandId and credentials (username and password).
	POST /sast/scans	Create a new scan and assign it to a specific project. Parameters include – isIncremental, isPublic, forceScan and comment.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Running Scheduled Scans

The following are the new additions to the Checkmarx API: public CxWSResponseRunID ScanWithSchedulingWithCron(string sessionId, CliScanArgs args, string cronString, long utcEpochStartTime, long utcEpochEndTime) This method is intended for generating a new scan job according to the CliScanArgs input, cron expression and start and end dates. Cron-Expressions Cron-Expressions are strings that are actually made up of seven sub-expressions, that describe individual details of the schedule. These sub-expression are separated with white-space, and represent:

1. Seconds

2. Minutes

3. Hours

4. Day-of-Month

5. Month

6. Day-of-Week

7. Year (optional field)

An example of a complete cron-expression is the string "0 0 12 ? * WED" - which means "every Wednesday at 12:00 pm".

Individual sub-expressions can contain ranges and/or lists. For example, the day of week field in the previous (which reads "WED") example could be replaces with "MON-FRI", "MON, WED, FRI", or even "MON-WED,SAT".

Wild-cards (the '*' character) can be used to say "every" possible value of this field. Therefore the '*' character in the "Month" field of the previous example simply means "every month". A '*' in the Day-Of-Week field would obviously mean "every day of the week".

All of the fields have a set of valid values that can be specified. These values should be fairly obvious - such as the numbers 0 to 59 for seconds and minutes, and the values 0 to 23 for hours. Day-of-Month can be any value 0-31, but you need to be careful about how many days are in a given month! Months can be specified as values between 0 and 11, or by using the strings JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV and DEC. Days-of- Week can be specified as values between 1 and 7 (1 = Sunday) or by using the strings SUN, MON, TUE, WED, THU, FRI and SAT.

The '/' character can be used to specify increments to values. For example, if you put '0/15' in the Minutes field, it means 'every 15 minutes, starting at minute zero'. If you used '3/20' in the Minutes field, it would mean 'every 20 minutes during the hour, starting at minute three' - or in other words it is the same as specifying '3,23,43' in the Minutes field.

The '?' character is allowed for the day-of-month and day-of-week fields. It is used to specify "no specific value". This is useful when you need to specify something in one of the two fields, but not the other.

The 'L' character is allowed for the day-of-month and day-of-week fields. This character is short-hand for "last", but it has different meaning in each of the two fields. For example, the value "L" in the day-of-month field means "the last day of the month" - day 31 for January, day 28 for February on non-leap years. If used in the day-of-week field by itself, it simply means "7" or "SAT". But if used in the day-of-week field after another value, it means "the last xxx day of the month" - for example "6L" or "FRIL" both mean "the last friday of the month". When using the 'L' option, it is important not to specify lists, or ranges of values, as you'll get confusing results.

The 'W' is used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: "the nearest weekday to the 15th of the month".

The '#' is used to specify "the nth" XXX weekday of the month. For example, the value of "6#3" or "FRI#3" in the dayof-week field means "the third Friday of the month".

Start and end times (utcEpochStartTime & utcEpochEndTime)

Those long type parameters are used to determine a repetitive (scheduled) scan start and end time. The default values are 0 (zero) in order to indicate that it should start now (though initiate a scan by the cron expression schedule) and never end / last forever.

Other (non default) values indicate the start and end time. They contain the second count from the Epoch (January 1st 1970) on UTC time. Note that the start time HAS to be in the future and that there must be at least one scan scheduled in between start and end time.

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

ScanWithSchedulingWithCron	PUT /sast/project/{projectId}/scheduling	Define specific project's scan scheduling settings. Parameters include - scheduleType and scheduleDays.
	GET /sast/scans/{id}	Get details of a specific scan. Returns status and stage of the scan.
	GET /sast/scanSettings/{projectId}	Get a specific project's scan settings. Returns preset and engine configuration of the scan.
	GET /sast/scansQueue	Get details of all scans in the scans queue. Returns wide-ranging scan information (e.g. stageDetails, engineId, languages, teamId, loc, origin, queuePosition, isIncremental, isPublic, origin, creation date, etc..).
	GET sast/scans?scanStatus={status}	Get all scans with a specific scan status (Scanning, Finished, Canceled or Failed).
	GET /sast/scans?last={numberOfLastScans}	Get all scans according to number of last scans.
	GET /sast/scans	Get all scans.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Getting Scan Status and Details

After running a scan, The API client can get the scan status and its details. To do this, the API will first need the scan's Run ID.

The obtained details include the scan's Scan ID, which can be subsequently used for commenting and reports.

CxSDKWebService.GetStatusOfSingleScan Method

```
public CxWSResponseScanStatus GetStatusOfSingleScan(

   string sessionID,

   string runId

);
```

Parameters

- **sessionID**: The current Session ID.
- **runId**: The scan's Run ID as obtained upon running the scan.

Return Value

CxWSResponseScanStatus, including:

- **.CurrentStatus**: The scan's status
- **.ScanID**: Once the scan is complete, **.ScanID** contains an ID that enables subsequent methods for commenting and reporting.

Example

To get the status and details of a scan with a known Run ID:

```
internal void Main(string [] args)

{

    String sessionID = args[0];

    CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



    //the scan's known identifier

    string scanRunID = "d95d56a2-2d8c-4e61-8146-0822213549f8";



    CxWSResponseScanStatus response =
cxSDKProxy.GetStatusOfSingleScan(sessionID, scanRunID);
```

```
    //store the scan's current state

    CurrentStatus = response.CurrentStatus;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetStatusOfSingleScan	GET /sast/scansQueue/{Id}	Get details of a specific scan in the scans queue.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Adding a Scan Comment

The API client can add a comment to scan results. If there is already a comment, it will be overwritten.

CxSDKWebService.UpdateScanComment Method

```
public CxWSBasicRepsonse UpdateScanComment(

   string sessionID,

   long ScanID,

   string Comment

);
```

Parameters

- **sessionID**: The current Session ID.
- **ScanID**: The ScanID of the scan to which to add the comment

Example

To add a comment to the results of a scan with known ID of **350**:

```
internal void Main(string [] args)

{

    String sessionID = args[0];

    CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



    long scanID = 350;



    //Create the comment

    string scanComment = string.Concat("The Scan ID of this scan is" ,
scanID.ToString());



    CxWSBasicRepsonse response
=cxSDKProxy.UpdateScanComment(sessionID, scanID, scanComment);

    UpdateSucceded = response.IsSuccesfull;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

UpdateScanComment	PATCH /sast/scans/{id}	Add a comment to a specific scan. Parameters include - comment.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Cancelling a Scan

The API client can cancel a scan in progress. The scan can be canceled while waiting in the queue or during a scan.

cxSDKProxy.CancelScan Method

```
public CxWSBasicRepsonse CancelScan(

   string sessionID,

   string scanRunID,

);
```

Parameters

- **sessionID**: The current Session ID.
- **runID**: The scan's Run ID as obtained upon running the scan.

Return Value

If there are any errors, the return value parameter “CxWSBasicRepsonse.IsSuccesfull” is set to false and the second parameter “CxWSBasicRepsonse.ErrorMessage” indicates the reason for failure.

Example

To cancel a scan:

```
internal void Main(string [] args)

{

    String sessionID = args[0];

    CxSDKWebService cxSDKProxy = new CxSDKWebService();


   //the scan unique run id

   string scanRunID = "d95d56a2-2d8c-4e61-8146-0822213549f8";


   //cancel the scan

   CxWSBasicRepsonse response = cxSDKProxy.CancelScan(sessionID,
scanRunID);

 }
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

CancelScan	PATCH /sast/scansQueue/{id}	Cancel a specific scan while still in the queue. Parameters include - status (cancelled).

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Deleting a Scan

The API client can delete requested scans. Scans that are currently running won't be deleted. If there's even a single scan that the user can't delete (due to security reasons) the operation fails and an error message is returned.

cxSDKProxy.DeleteScans Method

```
public CxWSBasicRepsonse DeleteScan(

   string sessionID,

   long scanIDs,

);
```

Parameters

- **sessionID**: The current Session ID.
- **scanIDs**: The scan unique id.

If there were any errors set CxWSBasicRepsonse.IsSuccesfull to false, specify the error in the CxWSBasicRepsonse.ErrorMessage

Return Value

If there are any errors, the return value parameter “CxWSBasicRepsonse.IsSuccesfull” is set to false and the second parameter “CxWSBasicRepsonse.ErrorMessage” indicates the reason for failure.

Example

To delete single or multiple scans:

```
internal void Main(string [] args)

   {

   String sessionID = args[0];


   CxSDKWebService cxSDKProxy = new CxSDKWebService();


   //the scans unique id
```

```
    long[] scanIDs = new long[]{1,2,3};


   //cancel the scan

   CxWSBasicRepsonse response = cxSDKProxy.DeleteScans(sessionID,
scanIDs);

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

DeleteScans	DELETE /sast/scans/{id}	Delete a specific scan.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Working with Scan Result Reports

This section includes SDK methods for working with scan result reports.

Generating a Report

The API client can generate a result report for a scan, by Scan ID.

CxSDKWebService.CreateScanReport Method

```
public CxWSCreateReportResponse CreateScanReport(
   string sessionID,
   CxWSReportRequest reportRequest
);
```

Parameters

- **sessionID**: The current Session ID
- **reportRequest**: An instance of class CxWSReportRequest, which includes the following fields:
  - **.ScanID**: The Scan ID of the scan results for which to generate a report.
  - **.Type**: The report output type.

Return Value

CxWSReportRequest, including:

.ID: Report ID to be used for tracking report generation process or for retrieving the report.

Example

To generate a result report for a scan with a known ID of **256**:

```
internal void Main(string [] args)

{

    String sessionID = args[0];

    CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



    CxWSReportRequest request = new CxWSReportRequest ();



    //set the report type to be PDF

    request.Type = CxWSReportType.PDF;



    //report should be for scan ID 256

    request.ScanID = 256;



    CxWSCreateReportResponse response =
cxSDKProxy.CreateScanReport(sessionID, request);

    ReportID =response.ID;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

CreateScanReport	POST /reports/sastScan	Generate a new scan report.

For more mapping information, refer to API Mapping (SOAP to REST). You can find a summary of our REST APIs here.

Getting Report Status

The API client can track the status of a report generation request.

CxSDKWebService.GetScanReportStatus Method

```
public CxWSReportStatusResponse GetScanReportStatus(
   string SessionID,
   long ReportID
);
```

Parameters

- **sessionID**: The current Session ID.
- **ReportID**: The report ID.

Return Value

CxWSReportStatusResponse, including:

- **.IsFailed** (boolean): If process failed, set to **true**
- **.IsReady** (boolean): If process ended, set to **true**

Example

To check the status of a report with a known report ID of **200**:

```
internal void Main(string [] args)

{

    String sessionID = args[0];

    CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



    int reportId = 200;



    CxWSReportStatusResponse response=
cxSDKProxy.GetScanReportStatus(sessionID, reportId);



    //if IsReady is true the creation process is done

    ReportReady = response.IsReady;



    //if IsFailed is true the creation process failed and the server
stopped the process

    GeneratingProcessFailed = response.IsFailed;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetScanReportStatus	GET /reports/sastScan/{Id}/status	Get the status of a generated report.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Getting a Report

Once a scan result report has been generated and the report is ready, the API client can retrieve the report's content.

CxSDKWebService.GetScanReport Method

```
public CxWSResponseScanResults GetScanReport(

   string SessionID,

   long ReportID

);
```

Parameters

- **sessionID**: The current Session ID.
- **ReportID**: The report ID.

Return Value

CxWSResponseScanResults, including:

- **.ScanResults** (byte array): The report content is the last scan log as file XML to download.
- **.containsAllResults** (boolean): **true** if report content hasn't been cut (due to configured maximal report size).

Example

To get the contents of a report with a known ID of **200**:

```
internal void Main(string [] args)

{

   String sessionID = args[0];

   CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();

   long repotID = 200;

   //ask for results of report with id 200

   CxWSResponseScanResults response =
cxSDKProxy.GetScanReport(sessionID, repotID);

   //get the report content as byte array

   ReportContent = response.ScanResults;

   //check if report content contains all scan results

   ContainAllResults = response.containsAllResults;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetScanReport	GET /reports/sastScan/{Id}	Get the specific report once generated.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Getting Group Information

The API client can get information on all groups related to the current user.

CxSDKWebService.GetAssociatedGroupsList Method

```
public CxWSResponseGroupList GetAssociatedGroupsList(

   string SessionID

);
```

Parameters

- **sessionID**: The current Session ID

Return Value

CxWSResponseGroupList.GroupList contains an array of group data.

Example

```
internal void Main(string [] args)

{

    String sessionID = args[0];

    CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



    CxWSResponseGroupList response =
cxSDKProxy.GetAssociatedGroupsList(sessionId);

    Teams = response.GroupList;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetAssociatedGroupsList	GET /auth/teams	Gets details of all teams.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Getting Available Encoding Options

The API client can get the list of available encoding options (for scan configuration).

CxSDKWebService.GetConfigurationSetList Method

```
public CxWSResponseConfigSetList GetConfigurationSetList(

   string SessionID

);
```

Parameters

- **sessionID**: The current Session ID

Return Value

Available encoding options.

Example

```
internal void Main(string [] args)

{

     String sessionID = args[0];

     CxSDKWebServiceSoapClient cxSDKProxy = new
CxSDKWebServiceSoapClient();



     CxWSResponseConfigSetList response =
cxSDKProxy.GetConfigurationSetList(sessionID);

     GetConfigurationsSucceded = response.IsSuccesfull;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetConfigurationSetList	GET /sast/engineConfiguration	Get details of all engine configurations.

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

Managing Users

To allow a user with the appropriate authorizations to retrieve the list of all users in the system (including users ID) and delete one or more of these users, the following methods were introduced to the SDK:

- GetAllUsers - allows the API client to get a list of all users in the system that are visible to the current user. For details, see GetAllUsers Method.
- DeleteUser - allows the API client to delete a user in the system. For details, see DeleteUser Method.

DeleteUser Method

CxSDKWebService.DeleteUser Method

```
public CxWSBasicResponse DeleteUser(string sessionID, int userID);
```

Parameters

- **sessionID**: The current Session ID
- **userID**: The user ID of the user to be deleted.

Return Value

CxWSBasicResponse, including:

- **IsSuccessful**: Indicates that the user has been successfully deleted, in which case the message "User deleted" is displayed.

Example

To delete a user:

```
internal void Main(string [] args)

{

    String sessionID = args[0];

int userId = int.Parse(args[1]);
CxSDKWebService cxSDKProxy = new CxSDKWebService();
```

```
CxWSResponseGroupList response =

    cxSDKProxy.DeleteUser(sessionId, userId);

if (response.IsSuccesfull)

{

Console.Writeline("User deleted");

}

else

{

Console.Writeline("User deletion failed");

}

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

DeleteUser	For future release

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.

GetAllUsers Method

CxSDKWebService.GetAllUsers Method

```
public CxWSResponseUserData GetAllUsers

(string sessionID

);
```

Parameters

sessionID: The current Session ID

Return Value

CxWSResponseUserData, which includes:

UserDataList: AN array that contains all visible users.

Example

To get the list of all visible users:

```
internal void Main(string [] args)

{

String sessionID = args[0];

CxSDKWebService cxSDKProxy = new CxSDKWebService();


      CxWSResponseGroupList response =

           cxSDKProxy.GetAllUsers(sessionId);

UserData[] users = response.UserDataList;

}
```

SOAP to REST Mapping

This section covers SOAP to REST migration and mapping of our legacy SOAP based SDK to the new REST APIs. It is recommended to use this reference only once CxSAST V8.8.0 is installed.

GetAllUsers	For future release

For more mapping information, refer to API Mapping (SOAP to REST). You can also find a summary of our REST APIs here.