**Synopsys Bridge CLI Guide**

# Contents

[Chapter 1. Overview](#_bookmark0) [4](#_bookmark0)

[Overview](#_bookmark1) [4](#_bookmark1)

[Support Matrix](#_bookmark2) [4](#_bookmark2)

[Files and Directories](#_bookmark3) [5](#_bookmark3)

[Download Synopsys Bridge](#_bookmark4) [5](#_bookmark4)

[Chapter 2. Synopsys Bridge CLI](#_bookmark5) [7](#_bookmark5)

[Using Synopsys Bridge CLI](#_bookmark6) [7](#_bookmark6)

[Passing Arguments using a JSON file](#_bookmark7) [7](#_bookmark7)

[Passing Arguments using the CLI](#_bookmark8) [9](#_bookmark8)

[Using Synopsys Bridge CLI with Polaris](#_bookmark9) [9](#_bookmark9)

[Using Synopsys Bridge CLI with Black Duck](#_bookmark13) [12](#_bookmark13)

[Using Synopsys Bridge CLI with Coverity](#_bookmark16) [15](#_bookmark16)

[Chapter 3. Synopsys Bridge CLI Reference](#_bookmark19) [19](#_bookmark19)

[Complete List of Synopsys Bridge Arguments](#_bookmark20) [19](#_bookmark20)

[Exit Codes](#_bookmark25) [33](#_bookmark25)

[Logging and Diagnostics](#_bookmark26) [33](#_bookmark26)

[Chapter 4. GitHub - Synopsys Action](#_bookmark27) [35](#_bookmark27)

[GitHub Prerequisites](#_bookmark28) [35](#_bookmark28)

[Using Synopsys GitHub Action for Polaris](#_bookmark29) [36](#_bookmark29)

[Using Synopsys GitHub Action for Black Duck](#_bookmark30) [37](#_bookmark30)

[Using Synopsys GitHub Action for Coverity Cloud Deployment with Thin Client](#_bookmark31) [41](#_bookmark31)

[Additional GitHub Configuration](#_bookmark32) [44](#_bookmark32)

[Chapter 5. GitLab – Synopsys Template](#_bookmark33) [45](#_bookmark33)

[GitLab Prerequisites](#_bookmark34) [45](#_bookmark34)

[Using the Synopsys GitLab Template with Polaris](#_bookmark35) [46](#_bookmark35)

[Using the Synopsys GitLab Template with Black Duck](#_bookmark36) [48](#_bookmark36)

[Using the Synopsys GitLab Template for Coverity Cloud Deployment with Thin Client](#_bookmark37) [51](#_bookmark37)

Contents | iii

[Additional GitLab Configuration](#_bookmark38) [54](#_bookmark38)

[Chapter 6. Azure DevOps - Synopsys Security Scan](#_bookmark39) [55](#_bookmark39)

[Azure Prerequisites](#_bookmark40) [55](#_bookmark40)

[Using Azure DevOps Extension with Polaris](#_bookmark41) [56](#_bookmark41)

[Using Azure DevOps Extension with Black Duck](#_bookmark42) [57](#_bookmark42)

[Using Azure DevOps Extension with Coverity Connect with Thin Client](#_bookmark43) [60](#_bookmark43)

[Additional Azure Configuration](#_bookmark44) [63](#_bookmark44)

[Chapter 7. Glossary](#_bookmark45) [65](#_bookmark45)

# Chapter 1. Overview

## Overview

Use any of these to integrate Static Analysis Security Testing (SAST) and Software Composition Analysis (SCA) into your CI/CD pipelines:

* Synopsys Bridge CLI
* Synopsys Action for GitHub
* Synopsys Template for GitLab
* Synopsys Security Scan for Azure DevOps With any of the integrations above you can:
* Scan when you merge code or on a pull request.
* Optionally create pull request comments when new issues are found.
* Optionally create a new pull request to automatically update vulnerable components.

#### Further information

* [Support Matrix *(on*](#_bookmark2)[*page 4)*](#_bookmark2)
* [Files and Directories *(on*](#_bookmark3)[*page 5)*](#_bookmark3)
* [Download Synopsys Bridge *(on*](#_bookmark4)[*page 5)*](#_bookmark4)

## Support Matrix

The table below outlines which Synopsys security tools are supported by Synopsys Bridge.

|  |  |  |
| --- | --- | --- |
| **Tool** | **Bridge Support?** | **Notes** |
| Polaris | Yes | Polaris users can use Synopsys Bridge CLI to automate SAST and/or SCA scans in their CI pipeline. Click here for SAST spe­ cific [system requirements](https://sig-product-docs.synopsys.com/bundle/coverity-docs/page/deploy-install-guide/topics/minimum_requirements.html). |
| Black Duck | Yes | Black Duck users can use Synop­ sys Bridge CLI to automate SCA scans in their CI pipeline. |

Synopsys Bridge CLI Guide | 1 - Overview | 5

|  |  |  |
| --- | --- | --- |
| **Tool** | **Bridge Support?** | **Notes** |
| Coverity Connect | Yes | Coverity users can use Synopsys Bridge CLI to automate SAST scans in their CI pipeline. Synop­ sys Bridge can be used with both an on-prem Coverity Connect as well as Coverity cloud deploy­ ment. Click here for [system re­](https://sig-product-docs.synopsys.com/bundle/coverity-docs/page/deploy-install-guide/topics/minimum_requirements.html) [quirements](https://sig-product-docs.synopsys.com/bundle/coverity-docs/page/deploy-install-guide/topics/minimum_requirements.html). |

Synopsys Bridge runs on the following operating systems:

##### OS System Requirements Notes

Linux 64-bit kernel, version 2.6.32+ with glibc 2.18 or later

Debian GNU is *not* supported

macOS OSX 11, 12, 13 macOS 11, 12 and 13 on Intel (M1 and M2 based Macs are not currently supported)

Windows x86\_64, Version 10 and 11 and Windows Server 2019 and 2022

Server Core is *not* supported

## Files and Directories

By default, the Synopsys Bridge Bridge writes logs and temporary files to <current\_working\_directory>/

.bridge. You may change this default directory by using the --home <directory\_path> option. The following files and directories are found under the Synopsys Bridge home directory:

* bridge.log
* diagnostics.json file with --diagnostics option. See [Logging and Diagnostics *(on*](#_bookmark26)[*page 33)*](#_bookmark26) for details.
* Adapter directories and the corresponding stdout and stderr log files.
* Additional temporary files.

## Download Synopsys Bridge

You can download the latest version of Synopsys Bridge from [Synopsys Artifactory](https://sig-repo.synopsys.com/artifactory/bds-integrations-release/com/synopsys/integration/synopsys-bridge/). Polaris users can also download Synopsys Bridge directly from the Polaris user interface:

Synopsys Bridge CLI Guide | 1 - Overview | 6

* Click **username** at the top right.
* Select **Accounts**.
* Select **Downloads**.
* Choose the appropriate package for your operating system.

To install, simply unzip and add synopsys-bridge executable to your PATH or use absolute PATH to

synopsys-bridge executable.

# Chapter 2. Synopsys Bridge CLI

## Using Synopsys Bridge CLI

Once you have synopsys-bridge executable installed, you are ready to use Synopsys Bridge to integrate SAST and/or SCA scans into your CI/CD pipeline.

You can run Synopsys Bridge in one of the following two ways:

1. By passing arguments through a JSON file.
2. By passing arguments on the command line.

For a complete list of exit codes returned by Synopsys Bridge, see the [Exit Code *(on*](#_bookmark25)[*page 33)*](#_bookmark25) table.

### Passing Arguments using a JSON file

Passing arguments using a JSON file greatly simplifies the command line and promotes reuse. Here are the steps:

1. Create an access token in the web interface of the Synopsys security product you are integrating with.
2. Use environment variable(s) to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Synopsys Bridge automatically picks up values passed through these variables.
   * Example: export BRIDGE\_POLARIS\_ACCESSTOKEN=*<POLARIS\_ACCESSTOKEN>*.
3. Pass the JSON file to Synopsys Bridge using the --input command line option.
4. Pass the Synopsys security product you are integrating with using the --stage option.

Here are the example commands::

export BRIDGE\_POLARIS\_ACCESSTOKEN=<POLARIS\_ACCESSTOKEN>

synopsys-bridge --stage polaris --input input.json



**Note:**

Depending on your OS, you will need to use appropriate mechanism to set environment variables.

Here is the input.json file:

{

"data": {

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 8

"polaris": { "application": {

"name": "<APPLICATION\_NAME>"

},

"project": {

"name": "<PROJECT\_NAME>"

},

"assessment": {

"types": ["SAST", "SCA"]

},

"serverUrl": "<SERVER\_URL>"

}

}

}



**Note:**

It is recommended that you save the JSON file at the root of the project directory being scanned. The JSON file can have any name as long as it has a .json extension.



**Note:**

You can use different JSON files for different use cases.

For a complete list of environment variables and command line arguments, see [Complete List of](#_bookmark20) [Synopsys Bridge Arguments](#_bookmark20) *[(on](#_bookmark20) [page 19)](#_bookmark20)*.

For tool specific information and examples, see:

* [Using Synopsys Bridge with Polaris *(on*](#_bookmark9)[*page 9)*](#_bookmark9)
* [Using Synopsys Bridge with Black Duck *(on*](#_bookmark13)[*page 12)*](#_bookmark13)
* [Using Synopsys Bridge with Coverity Connect *(on*](#_bookmark16)[*page 15)*](#_bookmark16)

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 9

### Passing Arguments using the CLI

You can also pass arguments on the command line as an alternative to passing arguments using a JSON file.

Here are the steps:

1. Create an access token in the web interface of the Synopsys security product you are integrating with.
2. Use environment variable(s) to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Synopsys Bridge automatically picks up values passed thru these variables.
   * Example: export BRIDGE\_POLARIS\_ACCESSTOKEN=*<POLARIS\_ACCESSTOKEN>*
3. Pass the necessary command line arguments as shown in the example below.

export BRIDGE\_POLARIS\_ACCESSTOKEN="<POLARIS\_ACCESSTOKEN>"

synopsys-bridge --stage polaris polaris.project.name="<PROJECT\_NAME>" \ polaris.application.name="<APPLICATION\_NAME>" \ polaris.assessment.types=SAST,SCA \ polaris.serverUrl="<POLARIS\_SERVERURL>"

For a complete list of environment variables and command line arguments, see [Complete List of](#_bookmark20) [Synopsys Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).

See Schema Resources And Extensions *(on page )* for Synopsys Bridge resources. For tool specific information and examples, see:

* [Using Synopsys Bridge with Polaris *(on*](#_bookmark9)[*page 9)*](#_bookmark9)
* [Using Synopsys Bridge with Black Duck *(on*](#_bookmark13)[*page 12)*](#_bookmark13)
* [Using Synopsys Bridge with Coverity Connect *(on*](#_bookmark16)[*page 15)*](#_bookmark16)

## Using Synopsys Bridge CLI with Polaris

As a Polaris customer, you can use Synopsys Bridge to automate SAST and SCA scanning in your CI pipeline.

You can use Synopsys Bridge to run Polaris scans in the following two ways:

* [Running Polaris scans with a JSON file *(on*](#_bookmark10)[*page 10)*](#_bookmark10)
* [Running Polaris scans on the command line *(on*](#_bookmark11)[*page 11)*](#_bookmark11)

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 10

In addition to running scans, you can also optionally configure Synopsys Bridge to create fix pull requests for SCA issues. Currently, only NPM is supported. For more information, see [Complete List of Synopsys](#_bookmark20) [Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).



**Note:**

As an alternative to Synopsys Bridge, you can also use [Synopsys Github Action *(on*](#_bookmark27)[*page 35)*](#_bookmark27), [Synopsys Template for GitLab *(on*](#_bookmark33)[*page 45)*](#_bookmark33) or [Synopsys Security Scan for Azure DevOps *(on*](#_bookmark39)[*page 55)*](#_bookmark39).

#### Running Polaris scans with a JSON file

Synopsys Bridge for Polaris uses Coverity for SAST scans and Black Duck for SCA scans under the hood. Depending on the task, you may need to pass additional SAST and SCA configurations.

Pass sensitive access token information using the BRIDGE\_POLARIS\_ACCESSTOKEN environment variable, and run Synopsys Bridge and pass the JSON file using the --input command line option.

Here is a command line example for Polaris:

export BRIDGE\_POLARIS\_ACCESSTOKEN=<POLARIS\_ACCESSTOKEN>

synopsys-bridge --stage polaris --input input.json

The above example uses the following:

* BRIDGE\_POLARIS\_ACCESSTOKEN environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product you are integrating with.

Here is the input.json file:

{

"data": {

"polaris": {

"application": {

"name": "*<APPLICATION\_NAME>*"

},

"project": {

"name": "*<PROJECT\_NAME>*"

},

"assessment": {

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 11

"types": ["SCA", "SAST"]

},

"serverUrl": "<*POLARIS\_URL>*"

}

}

}

The above JSON file uses the following:

* polaris.serverUrl for Polaris URL.
* polaris.application.name for Polaris Application to use. Note that the specified application must exist on Polaris with appropriate entitlements.
* polaris.project.name for Polaris Project to use. Note that the specified project must exist on Polaris. If you set polaris.onboarding to true, Synopsys Bridge will automatically create the project in case it doesn’t yet exist on Polaris.
* polaris.assessment.types specifies the type of scan to be run: SAST or SCA or SAST,SCA.

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Polaris, refer to the Polaris specific resources page under Schema Resources and Extensions *(on page*

*)*.

For a complete list of environment variables and command line arguments, see [Complete List of](#_bookmark20) [Synopsys Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).

For additional SAST-specific details, see [Additional SAST configuration requirements *(on*](#_bookmark12)[*page 12)*](#_bookmark12).

#### Running Polaris scans on the command line

Instead of using a JSON file, you can pass all arguments via the command line. Here is a command line example for Polaris:

export BRIDGE\_POLARIS\_ACCESSTOKEN=*<POLARIS\_ACCESSTOKEN>*

synopsys-bridge --stage polaris polaris.project.name="*<PROJECT\_NAME>*" \ polaris.application.name="*<APPLICATION\_NAME>*" \ polaris.assessment.types=SAST,SCA \

polaris.serverUrl="*<SERVERURL>"*

The above example uses the following:

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 12

* BRIDGE\_POLARIS\_ACCESSTOKEN environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use.
* polaris.serverUrl for Polaris URL.
* polaris.application.name for Polaris Application to use. Note that the specified application must exist on Polaris with appropriate entitlements.
* polaris.project.name for Polaris Project to use. Note that the specified project must exist on Polaris. If you set polaris.onboarding to true, Synopsys Bridge will automatically create the project in case it doesn’t yet exist on Polaris.
* polaris.assessment.types specifies the type of scan to be run: SAST or SCA or SAST,SCA.

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Polaris, refer to the Polaris specific resources page underSchema Resources and Extensions *(on page*

*)*.

For a complete list of environment variables and command line arguments, see [Complete List of](#_bookmark20) [Synopsys Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).

For additional SAST-specific details, see [Additional SAST configuration requirements *(on*](#_bookmark12)[*page 12)*](#_bookmark12).

#### Additional SAST configuration requirements

A coverity.yml configuration file is required for

* Static analysis of compiled languages like C/C++, C# and Java.
* Optimizing static analysis when results are unsatisfactory.

Certain Coverity Connect scans on Polaris require configuration of additional capture settings using a coverity.yaml file. See [Configuring Coverity Thin Client for use with Synopsys Bridge and Polaris](https://polaris.synopsys.com/developer/default/documentation/t_cov-thin-client) in the *Polaris Developer Portal* for more information.

## Using Synopsys Bridge CLI with Black Duck

As a Black Duck customer, you can use Synopsys Bridge to automate SCA scanning in your CI pipeline.

You can use Synopsys Bridge with Black Duck in the following two ways:

* [Running Black Duck scans with a JSON file *(on*](#_bookmark14)[*page 13)*](#_bookmark14)
* [Running Black Duck scans on the command line *(on*](#_bookmark15)[*page 14)*](#_bookmark15)

In addition to running full scans, you can also optionally configure Synopsys Bridge to perform the following:

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 13

* Scan pull requests
* Add comments to pull requests
* Create fix pull requests (NPM only)



**Note:**

As an alternative to Synopsys Bridge, you can also use [Synopsys GitHub Action *(on*](#_bookmark27)[*page 35)*](#_bookmark27), [Synopsys Template for GitLab *(on*](#_bookmark33)[*page 45)*](#_bookmark33) or [Synopsys Security Scan for Azure DevOps *(on*](#_bookmark39)[*page 55)*](#_bookmark39).

#### Running Black Duck scans with a JSON file

Pass sensitive information such as username and token using environmental variables, and run Synopsys Bridge and pass the JSON file using the --input command line option.

Here is an command line example for Black Duck:

export **BRIDGE\_BLACKDUCK\_TOKEN**=*<BLACKDUCK\_TOKEN>*

synopsys-bridge --stage blackduck --input input.json

The above example uses the following:

* BRIDGE\_BLACKDUCK\_TOKEN environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use.

Here is the input.json file:

{

"data": {

"blackduck": {

"url": *<BLACKDUCK\_URL>*,

"scan": {

"full": true,

"failure": {

"severities": ["CRITICAL"]

}

}

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 14

}

}

}

The above JSON file uses the following:

* blackduck.url for Black Duck URL.
* blackduck.scan.full should be set to true so that a full/intelligent scan is run by Synopsys Bridge. For pull request scans, this should be set to false.
* blackduck.scan.failure.severities is a list of severities used by Synopsys Bridge to decide if the CI pipeline should be failed or not.

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Polaris, refer to the Polaris specific resources page under Schema Resources and Extensions *(on page*

*)*.

For a complete list of environment variables and command line arguments, see [Complete List of](#_bookmark20) [Synopsys Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).

#### Running Black Duck scans on the command line

Instead of using a JSON file, you can pass all arguments on the command line. Here is a command line example for Black Duck:

export BRIDGE\_BLACKDUCK\_TOKEN=*<BLACKDUCK\_TOKEN>*

synopsys-bridge --stage blackduck \ blackduck.url=*<BLACKDUCK\_URL>* \ blackduck.scan.failure.severities=CRITICAL,HIGH \ blackduck.scan.full=true

The above example uses the following:

* BRIDGE\_BLACKDUCK\_TOKEN> environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use.
* blackduck.url for Black Duck URL.
* blackduck.scan.full should be set to true so that a full/intelligent scan is run by Synopsys Bridge. For pull request scans, this should be set to false.
* blackduck.scan.failure.severities is a comma separated list of severities that is used by Synopsys Bridge to decide if the CI pipeline should be failed or not.

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 15

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Polaris, refer to the Polaris specific resources page underSchema Resources and Extensions *(on page*

*)*.

For a complete list of environment variables and command line arguments, see [Complete List of](#_bookmark20) [Synopsys Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).

## Using Synopsys Bridge CLI with Coverity

As a Coverity customer, you can use Synopsys Bridge to automate SAST scanning in your CI/CD pipeline.



**Note:**

You can use Synopsys Bridge with both on-prem Coverity Connect as well as Coverity cloud deployment. Details below.

You can integrate Synopsys Bridge with Coverity in the following ways:

* [Running Coverity scans using a JSON file *(on*](#_bookmark17)[*page 15)*](#_bookmark17)
* [Running Coverity Connect scans using the command line *(on*](#_bookmark18)[*page 17)*](#_bookmark18)

In addition to running scans, you can also optionally configure Synopsys Bridge to add comments to pull requests. For more information, see [Complete List of Synopsys Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).



**Note:**

As an alternative to Synopsys Bridge, you can also use [Synopsys Github Action *(on*](#_bookmark27)[*page 35)*](#_bookmark27), [Synopsys Template for GitLab *(on*](#_bookmark33)[*page 45)*](#_bookmark33) or [Synopsys Security Scan for Azure DevOps *(on*](#_bookmark39)[*page 55)*](#_bookmark39).

#### Running Coverity scans using a JSON file

Pass sensitive information such as username and password using environmental variables, and run Synopsys Bridge and pass the JSON file using the --input command line option.

Here is the example command line:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME="*<COV\_USER>*"

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD="*<COVERITY\_PASSPHRASE>*"

synopsys-bridge --stage connect --input input.json

Here is an example the input.json file that you can use with on-prem Coverity Connect:

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 16

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*", "project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

"view": "*<View Name / Id>*"

}

},

"local": true

}

}

}

Here is an example input.json file that you can use with Coverity cloud deployment:

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*", "project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 17

"view": "*<View Name / Id>*"

}

}

}

}

}

The above examples use the following:

* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME and BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD environment variables to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage to specify the Synopsys security product in use.
* coverity.connect.url for Coverity Connect URL.
* coverity.connect.project.name for project on Coverity Connect. Must exist on the server.
* overity.connect.stream.name for stream on Coverity Connect. Must exist on the server.
* coverity.connect.policy.view for policy view to be used by Synopsys Bridge to decide if the CI pipeline should be failed or not.
* Coverity.local to let Synopsys bridge know if this is an on-prem Coverity Connect or a Coverity cloud deployment.

#### Running Coverity Connect scans using the command line

Instead of using a JSON file, you can pass arguments on the command line.

Here are the example commands that you can use with on-prem Coverity Connect:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME=<COV\_USER>

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD=<COVERITY\_PASSPHRASE>

synopsys-bridge --stage bridge \ coverity.connect url=<COVERITY\_URL> \

coverity.connect.project.name=<COVERITY\_PROJECT> \ coverity.connect.stream.name=<COVERITY\_STREAM> \ coverity.connect.policy.view=<COVERITY\_VIEW\_NAME> \

coveriy.local=true

Here are the example commands that you can use with Coverity cloud deployment:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME=<COV\_USER>

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD=<COVERITY\_PASSPHRASE>

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 18

synopsys-bridge --stage bridge \ coverity.connect url=<COVERITY\_URL> \ coverity.connect.project.name=<COVERITY\_PROJECT> \ coverity.connect.stream.name=<COVERITY\_STREAM> \

coverity.connect.policy.view=<COVERITY\_VIEW\_NAME>

The above examples use the following:

* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME and BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD environment variables to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage to specify the Synopsys security product in use.
* coverity.connect.url for Coverity Connect URL.
* coverity.connect.project.name for project on Coverity Connect to be used. Must exist on the server.
* coverity.connect.stream.name for stream on Coverity Connect to be used. Must exist on the server.
* coverity.connect.policy.view for policy view to be used by Synopsys Bridge to decide if the CI pipeline should be failed or not.
* Coverity.local to let Synopsys bridge know if this is an on-prem Coverity Connect or a Coverity cloud deployment.

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Polaris, refer to the Polaris specific resources page under Schema Resources and Extensions *(on page*

*)*.

For more details, see the [Complete List of Synopsys Bridge Arguments *(on*](#_bookmark20)[*page 19)*](#_bookmark20).

# Chapter 3. Synopsys Bridge CLI Reference

## Complete List of Synopsys Bridge Arguments

This page lists all the arguments that Synopsys Bridge supports. Arguments can be passed thru environment variables, command line or a JSON file.



**Note:**

We recommend that you pass sensitive information such as access tokens using environment variables.

For a list of arguments that are common to all Synopsys security products, refer to [Universal Synopsys](#_bookmark21) [Bridge Arguments *(on*](#_bookmark21)[*page 19)*](#_bookmark21) below.

For product specific arguments, refer to the product specific section below:

* [Polaris *(on*](#_bookmark22)[*page 20)*](#_bookmark22)
* [Black Duck *(on*](#_bookmark23)[*page 23)*](#_bookmark23)
* [Coverity Connect *(on*](#_bookmark24)[*page 29)*](#_bookmark24)

#### Universal Synopsys Bridge Arguments

These arguments can be passed on the command line, but not as part of a JSON file.

**Command**

**Description**

**Required?**

synopsys-bridge

Command to invoke Synopsys Bridge.

Yes

--stage

The --stage command specifies a group Yes of adapters to run (such as --stage po•

laris).

--input

The --input command loads a JSON file Required for inputting a JSON containing common arguments to run files.

scans

--help

Shows the help file for Synopsys Bridge. No

--json-log

Outputs JSON format logs. See [Logging](#_bookmark26) No

[and Diagnostics *(on*](#_bookmark26)[*page 33)*](#_bookmark26).

--json-log-file

Outputs JSON format logs in the

bridge.log file in the Synopsys Bridge

No

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 20

Enables debug logs under the Synopsys No Bridge home directory. Creates a diagnos• tics.json file containing the final state

data inside the Synopsys Bridge home di­ rectory, but masking sensitive information like tokens and passwords. See [Logging](#_bookmark26)

[and Diagnostics *(on*](#_bookmark26)[*page 33)*](#_bookmark26).

--diagnostics

No

Turns on verbose logging.

-- verbose

No

Specifies a schema to load

-- schema

Specifies a specific version of Synopsys No

Bridge to run.

-- version

No

Sets a home directory.

-- home

**Required?**

**Description**

home directory. See [Logging and Diag­](#_bookmark26) [nostics *(on*](#_bookmark26)[*page 33)*](#_bookmark26).

**Command**

#### Polaris

##### Arguments to Pass

**Argument Input Mode Required Notes**

**Command Line Argu­ ment**

**Environment Variable JSON field**

Access token polaris.ac•

cesstoken

Server URL po•

BRIDGE\_POLARIS\_ACCESS• TOKEN

BRIDGE\_POLARIS\_•

polaris.ac• cesstoken

po•

Yes For security reasons, it is recommend­ ed that you pass this as an environ­ ment variable.

Yes Polaris server

laris.serverurlSERVERURL

laris.serverurl

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 21

##### Argument Input Mode Required Notes

URL.

Application Name

polaris.ap• plication•

.name

BRIDGE\_POLARIS\_APPLI• CATION\_NAME

polaris.ap• plication•

.name

Yes Application must be creat­ ed on Polaris, and have right entitlements.

Project Name po•

laris.project•

.name

BRIDGE\_POLARIS\_•

PROJECT\_NAME

po• laris.project•

.name

Yes. The specified project must exist on Polaris. If you set polaris.onboarding to true, Synopsys Bridge will automatically create the project in case it doesn’t yet exist on Polaris

Assessment Type

Tool Install Di­ rectory

Auto Create Projects

polaris.as• sessment•

.types

tool.install•

.directory

polaris.on• boarding

BRIDGE\_POLARIS\_ASSESS• MENT\_TYPES

BRIDGE\_TOOL\_INSTALL\_• DIRECTORY

BRIDGE\_POLARIS\_ON• BOARDING

polaris.as• sessment•

.types

tool.install•

.directory

polaris.on• boarding

Yes Comma sep­ arated values. Accepted val­ ues SAST or SCA or SAST, SCA.

No Directory to which Bridge downloads the underlying scan tools.

Defaults to

*<User>*/.bridge.

No If set to true, Bridge will at­ tempt to cre­ ate the project on Polaris if it does not ex­

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 22

##### Argument Input Mode Required Notes

ist. Default is

false.

Polaris Triage po•

laris.triage

BRIDGE\_POLARIS\_TRIAGE po•

laris.triage

No If you are en­ titled to the Auto-Triage feature on Po­ laris, you can use this op­

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 23

##### Argument Input Mode Required Notes

tion to enable the feature. Possible val­ ues are RE•

NOT\_• REQUIRED and NOT\_ENTITLED.

QUIRED,

##### JSON Input

Here is a sample input.json file that can be used with Polaris:

{

"data": {

"polaris": {

"application": {

"name": "*<Application Name>*"

},

"project": {

"name": "*<Project Name>*"

},

"assessment": {

"types": ["SCA", "SAST"]

},

"serverUrl": "<Polaris URL>"

}

}

}

Here are the commands to run:

export BRIDGE\_POLARIS\_ACCESSTOKEN=<POLARIS\_ACCESSTOKEN>

synopsys-bridge --stage Polaris --input input.json

#### Black Duck

The base command to run the scan:

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 24

synopsys-bridge --stage blackduck

##### Arguments to Pass

**Argument Input Mode**

**Re­ quired**

**Notes**

Black Duck URL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Command Line Argu­**  **ment** | **Environment Variable** | **JSON field** |  |
| URL | blackduck.url | BRIDGE\_BLACKDUCK\_URL | blackduck.url | Yes |
| Token | blackduck.to• ken | BRIDGE\_BLACKDUCK\_TOKEN | blackduck.to• ken | Yes |
| Full scan | blackduck.s• can.full | BRIDGE\_BLACKDUCK\_SCAN\_• FULL | blackduck.s• can.full | No |

Black Duck Access token

Performs a full/in­ telligent scan when set to true. Re­ quired and used for scanning based on SCM push events.

Install Di­ rectory

blackduck.in• stall.direc• tory

BRIDGE\_BLACKDUCK\_INSTAL• L\_DIRECTORY

blackduck.in• No

stall.direc• tory

Performs a rapid scan when set to false . Required for SCM pull request events.

true or false. (De­ fault: false).

Path to directory where detect.jar resides.

Default: <

$HOME>/.bridge/ blackduck

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 25

##### Argument Input Mode

**Re­ quired**

**Notes**

Failure severities

blackduck.s• can.failure•

.severities

BRIDGE\_BLACKDUCK\_SCAN\_• FAILURE\_SEVERITIES

blackduck.s• No

can.failure•

.severities

Used by Bridge to determine whether to break the build or not.

If provided, Bridge will break the build and returns exit code.

Create fix pull re­ quests

blackduck.au• tomation.fix• pr

BRIDGE\_BLACKDUCK\_AU• TOMATION\_FIXPR

blackduck.au• No

tomation.fix• pr

If set to true, Bridge creates fix pull re­ quests for vulner­ able direct depen­ dencies. (Default: false)

Note: Currently only NPM is supports.

Note: Requires SCM information includ­ ing token as docu­ mented in section SCM Information needed for “Cre­ ating Fix Pull Re­ quests” feature be­ low.

Add com­ ments to pull re­ quests

blackduck.au• tomation•

.prcomments

BRIDGE\_BLACKDUCK\_AU• TOMATION\_PRCOMMENTS

blackduck.au• No

tomation•

.prcomments

If set to true , Bridge adds com­ ments to pull re­ quests for new is­

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 26

##### Argument Input Mode

**Re­ quired**

**Notes**

sues introduced in the pull request.

Note: Requires Rapid Scan to be run on pull require events. This argument is ignored if full scan is run.

Note: Requires SCM information includ­ ing token as docu­ mented in section SCM Information needed for "Adding Comments to Pull Requests" feature below.

##### SCM Information needed for “Creating Fix Pull Requests” feature

To use this feature, you must pass the following SCM arguments.

##### SCM Argument Input Mode

**Re­ quired**

**Command Line Ar­ gument**

**Environment Variable JSON Field**

GitHub User Token github.user.token BRIDGE\_GITHUB\_USER\_TOKEN github.user.token Yes

Repository Name

github.repository•

.name

BRIDGE\_GITHUB\_REPOSITO• RY\_NAME

github.repository•

.name

Yes

Branch Name github.repository•

.branch.name

BRIDGE\_GITHUB\_REPOSITO• RY\_BRANCH\_NAME

github.repository­

.branch.name

Yes

Repository Owner

github.repository•

.owner.name

BRIDGE\_GITHUB\_REPOSITO• RY\_OWNER\_NAME

github.repository•

.owner.name

Yes

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 27

##### SCM Argument Input Mode

**Re­ quired**

GitLab

GitLab API URL

gitlab.api.url BRIDGE\_GITLAB\_URL gitlab.api.url Yes

User Token gitlab.user.token BRIDGE\_GITLAB\_USER\_TOKEN gitlab.user.token Yes

Repository Name

gitlab.repository•

.name

BRIDGE\_GITLAB\_REPOSITO• RY\_NAME

gitlab.repository•

.name

Yes

Branch Name gitlab.repository•

.branch.name

BRIDGE\_GITLAB\_REPOSITO• RY\_BRANCH\_NAME

gitlab.repository•

.branch.name

Yes

Azure

Azure API URL

azure.api.url BRIDGE\_AZURE\_API\_URL azure.api.url Yes

User Token azure.user.token BRIDGE\_AZURE\_USER\_TOKEN azure.user.token Yes

Organization Name

azure.organization•

.name

TION\_NAME

azure.organization•

.name

BRIDGE\_AZURE\_ORGANIZA•

Yes

Project Name azure.project.name

NAME

BRIDGE\_AZURE\_PROJECT\_•

azure.project.name Yes

Repository Name

azure.repository•

.name

RY\_NAME

azure.repository•

.name

BRIDGE\_AZURE\_REPOSITO•

Yes

Branch Name azure.repository•

.branch.name

RY\_BRANCH\_NAME

azure.repository•

.branch.name

BRIDGE\_AZURE\_REPOSITO•

Yes

Pull Request Number

azure.repository•

.pull.number

RY\_PULL\_NUMBER

azure.repository•

.pull.number

BRIDGE\_AZURE\_REPOSITO•

Yes

##### SCM Information needed for “Adding Comments to Pull Requests” feature

**SCM Argument Input Mode**

**Re­ quired**

**Command Line Ar­ gument**

**Environment Variable JSON Field**

GitHub User Token github.user.token BRIDGE\_GITHUB\_USER\_TOKEN github.user.token Yes

Repository Name

github.repository•

.name

BRIDGE\_GITHUB\_REPOSITO• RY\_NAME

Yes

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 28

##### SCM Argument Input Mode

Branch Name

BRIDGE\_GITHUB\_REPOSITO•

github.repository•

##### Re­ quired

Yes

github.repository•

Repoository Owner

Pull Request Number

.branch.name

.owner.name

github.repository•

github.repository•

.pull.number

RY\_BRANCH\_NAME

RY\_OWNER\_NAME

BRIDGE\_GITHUB\_REPOSITO•

BRIDGE\_GITHUB\_REPOSITO•

RY\_PULL\_NUMBER

.branch.name

.owner.name

github.repository•

github.repository•

.pull.number

Yes

Yes

GitLab

GitLab API URL

gitlab.api.url BRIDGE\_GITLAB\_URL gitlab.api.url Yes

User Token gitlab.user.token BRIDGE\_GITLAB\_USER\_TOKEN gitlab.user.token Yes

Azure

Repository Name

Branch Name

Pull Request Number

Azure API URL

User Token

Organization Name

Project Name

Repository Name

Branch Name

gitlab.repository•

.name

gitlab.repository•

.branch.name

gitlab.repository•

.pull.number azure.api.url

azure.user.token azure.organization•

.name

azure.project.name

azure.repository•

.name

azure.repository•

.branch.name

BRIDGE\_GITLAB\_REPOSITO•

RY\_NAME

BRIDGE\_GITLAB\_REPOSITO•

RY\_BRANCH\_NAME

BRIDGE\_GITLAB\_REPOSITO•

RY\_PULL\_NUMBER BRIDGE\_AZURE\_API\_URL

BRIDGE\_AZURE\_USER\_TOKEN

BRIDGE\_AZURE\_ORGANIZA• TION\_NAME

BRIDGE\_AZURE\_PROJECT\_• NAME

BRIDGE\_AZURE\_REPOSITO• RY\_NAME

BRIDGE\_AZURE\_REPOSITO• RY\_BRANCH\_NAME

gitlab.repository•

.name

gitlab.repository•

.branch.name

gitlab.repository•

.pull.number azure.api.url

azure.user.token azure.organization•

.name

azure.project.name

azure.repository•

.name

azure.repository•

.branch.name

Yes

Yes

Yes

Yes

Yes Yes

Yes

Yes

Yes

##### JSON Input

Here is a sample input.json file that can be used with Black Duck:

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 29

{

"data": {

"blackduck": {

"url": *<BlackDuck url>*, "scan": {

"full": true,

"failure": {

"severities": ["CRITICAL"]

}

}

}

}

}

Here are the commands to run:

export BRIDGE\_BLACKDUCK\_TOKEN=<BLACKDUCK\_TOKEN>

synopsys-bridge --stage blackduck --input input.json

#### Coverity Connect

##### Argu­ ment

Coveri­ ty URL

##### Command Line Argu­ ment

coveri• ty.con• nect.url

##### Input Mode

**Environment Variable**

BRIDGE\_COV• ERITY\_CON• NECT\_URL

##### JSON

**field**

coveri• ty.con• nect.url

##### Re­ quired

Yes

##### Notes

User Name

Pass­ word

coveri• ty.con• nect•

.user•

.name

coveri• ty.con• nect•

BRIDGE\_COV• ERITY\_CON• NECT\_USER\_• NAME

BRIDGE\_COV• ERITY\_CON•

coveri• ty.con• nect•

.user•

.name

coveri• ty.con• nect•

Yes For security reasons it is recommended to pass this as an environmental variable.

Yes For security reasons it is recommended to pass this as an environmental variable.

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 30

##### Argu­ ment

**Input Mode**

**Re­ quired**

**Notes**

Project Name

Stream Name

.user•

.password

coveri• ty.con• nect•

.project•

.name

coveri­ ty.connec­ t.stream­

.name

NECT\_USER\_• PASSWORD

BRIDGE\_• COVERITY\_• CONNECT\_• PROJECT\_• NAME

BRIDGE\_• COVERITY\_• CONNECT\_• STREAM\_NAME

.user•

.password

coveri•

ty.con•

nect•

.project•

.name

coveri•

ty.con•

nect•

.stream•

.name

Yes Project must exist on Coverity Instance

Yes Stream must exist on Coverity Instance.

View coveri• ty.con• nect.pol• icy.view

BRIDGE\_COV• ERITY\_CON• NECT\_POLI• CY\_VIEW

No

ty.con•

nect.pol• icy.view

coveri•

Coverity platform's view name/ID.

Bridge will break the build if issues are found in the view provided by user and returns [exit code](#_bookmark25) [*(on*](#_bookmark25)[*page 33)*](#_bookmark25).

Add com­ ments to pull re­ quests

coveri• ty.con• nect.au• tomation•

.prcom• ment

BRIDGE\_COV• ERITY\_CON• NECT\_AU• TOMATION\_• PRCOMMENT

No

ty.con•

nect.au• tomation•

.prcom•

coveri•

ment

If set to true , Bridge adds comments to pull re­ quests for new issues introduced in the pull re­ quest.

Requires Rapid Scan to be run on pull require events. This argument is ignored if full scan is run.

Note: Requires SCM information including to­ ken as documented in section SCM Information needed for “Adding Comments to Pull Requests” feature.

Install directo­ ry

cover• ity.in• stall.di• rectory

BRIDGE\_COV• ERITY\_INS• TALL\_DIREC• TORY

cover• No

ity.in•

stall.di•

rectory

Path to directory where coverity resides. De­ fault: <$HOME>/.bridge/coverity.

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 31

##### Argu­ ment

local analy­ sis

coveri• ty.local

##### Input Mode

BRIDGE\_COV• ERITY\_LOCAL

coveri• ty.local

##### Re­ quired

No

##### Notes

To use Synopsys Bridge with on-prem Coveri­ ty Connect, set this to true. When set to true, Bridge will download full analysis kit and will perform capture and analysis locally.

With Coverity cloud deployments, Synopsys us­ es Thin Client and this option should be set to false.

Default: false.



**Note:**

To use Synopsys Bridge with on-prem Coverity Connect, you must set the “Coverity.local” to true as described above.

Here is a sample input.json file that can be used with Coverity Cloud:

:

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*",

"project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

"view": "*<View Name / Id>*"

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 32

},

"automation": { "prcomment" : false

}

}

}

}

}

Here is a sample input.json file that can be used with on-prem Coverity Connect:

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*",

"project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

"view": "*<View Name / Id>*"

},

"automation": { "prcomment" : false

}

},

“local” : true}

}

}

}

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 33

Here are the commands to run:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME=<COV\_USER>

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD=<COVERITY\_PASSPHRASE>

synopsys-bridge --stage connect --input input.json

## Exit Codes

Synopsys Bridge returns the following exit codes depending on execution results. Any exit code other than 0 should be seen as a build-breaking condition in your CI/CD platform.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Code** | **Code Name** | **Description** |
| 0 |  | Normal | Synopsys Bridge exited without any errors. |
| 1 |  | UndefinedError | Undefined errors. Review the log file for details. |
| 2 |  | AdapterError | Synopsys Bridge received a non-0 exit code from an inter­ nal adapter. Review the log file for details. |
| 3 |  | ShutdownFailed | Synopsys Bridge failed to shut itself down after running the command. Review the log for details. |
| 8 |  | BridgeBuildBreak | The config option bridge.break is set to true but Synopsys Bridge is unable to enforce this. As a workaround, create a simple script to call Synopsys Bridge and implement build break logic in your script. |
| 9 |  | StartupFailed | Failed to initiate Synopsys Bridge. Review the log for de­ tails. |

## Logging and Diagnostics

Synopsys Bridge offers multiple logging and diagnostic options. By default, logs are written to

<current\_working\_directory>/.bridge directory. User can change this default location by passing the -- home <directory\_path> option.

#### Logging

Synopsys Bridge offers the following JSON format logging options:

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 34

* Pass --json-log to output JSON format logs to console.
* Pass --json-log-file to enable JSON format logs in the bridge.log file in the Synopsys Bridge home directory .

#### Diagnostics

To enable Synopsys Bridge diagnostics mode, pass a --diagnostics command line option. With this option set, Synopsys Bridge:

* writes additional diagnostics information to bridge.log.
* passes diagnostics related options to underlying tools so that they create logs under the Synopsys Bridge home directory.
* writes execution state date to diagnostics.json file under the Synopsys Bridge home directory.

# Chapter 4. GitHub - Synopsys Action

The Synopsys GitHub Action can be used to integrate Synopsys security testing into your CI pipeline. You can download Synopsys GitHub Action directly from the GitHub Marketplace link at: [https://github.com/](https://github.com/marketplace/actions/synopsys-action) [marketplace/actions/synopsys-action](https://github.com/marketplace/actions/synopsys-action).

By including and configuring the Synopsys Action in your *workflow*.yml file, you can quickly integrate Synopsys security products into your CI pipeline. We recommend using GitHub secrets for sensitive data like access tokens.

For more information, see:

* [GitHub Prerequisites *(on*](#_bookmark28)[*page 35)*](#_bookmark28)
* [Using Synopsys GitHub Action for Polaris *(on*](#_bookmark29)[*page 36)*](#_bookmark29)
* [Using Synopsys GitHub Action for Black Duck *(on*](#_bookmark30)[*page 37)*](#_bookmark30)
* [Using Synopsys GitHub Action for Coverity Cloud Deployment with Thin Client *(on*](#_bookmark31)[*page 41)*](#_bookmark31)
* [Additional GitHub Configuration *(on*](#_bookmark38)[*page 54)*](#_bookmark38)

## GitHub Prerequisites

Before configuring Synopsys Action into your workflow, you must meet the following prerequisites:

#### GitHub Runner Setup

* Runners are the machines that execute jobs in a GitHub Actions workflow. To use GitHub runners in your project, GitHub Actions must be enabled for a repository/organization settings in order for required workflows to run (**Repository Settings** → **SelectActions** → **General** → **Actions permissions**).
* GitHub runner can be Self-hosted or GitHub-hosted. For installing Self-hosted runners, see [Self- hosted runners](https://docs.github.com/en/actions/hosting-your-own-runners). For installing GitHub-hosted runners, see [GitHub-hosted runners](https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners).

#### Configure GitHub Secrets

Sensitive data such as access tokens, user names, passwords and even URLs must be configured using GitHub secrets (**GitHub** → **Project** → **Settings** → **Secrets and Variables** → **Actions**).

#### Configure GitHub Token

github\_token is required as input when running Black Duck Fix PR, Black Duck/Coverity PR Comment. There are two different types of tokens that can be passed to github\_token:

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 36

* Token can be GitHub specified secrets.GITHUB\_TOKEN with required workflow read and write permissions (**GitHub** → **Project** → **Settings** → **Actions** → **General** → **Workflow Permissions**). It will be created by GitHub at start of each workflow run.
* If you need a token that requires permissions that aren't available in the secrets.GITHUB\_TOKEN, create a Personal Access Token (PAT) with required scopes (**Select Profile Photo** → **Settings** → **Developer Settings** → **Personal access tokens**). For more information, see [Granting Additional Permissions](https://docs.github.com/en/actions/security-guides/automatic-token-authentication#granting-additional-permissions). PAT must have repo and api scope to perform Black Duck Fix PR or Black Duck/ Coverity PR Comment.

#### Create workflow

Create a new workflow (**GitHub** → **Project** → **Actions** → **New Workflow** → **Setup a workflow yourself**) and configure he required fields. Push those changes and GitHub runner will initiate the workflow which can be seen on the **Actions** tab on main page of the repository.

## Using Synopsys GitHub Action for Polaris

Before running a pipeline using the Synopsys GitHub Action with Polaris, you must set the appropriate applications and entitlements in your Polaris environment.

Using Synopsys Action, you can perform scans on push events to main branches. Pull request scanning is currently not supported for Polaris.

Add the following code block to your existing *workflow*.yml file in your .github/workflows directory. (If you need to create a workflow, go to the repository you're integrating with Polaris on the GitHub UI, click the **Actions** tab at the top, then click **New Workflow**.)

Below is an example of a *workflow*.yml file configured for Polaris.

name: polaris-sig-action on:

push:

branches: [ main, master, develop, stage, release ] workflow\_dispatch:

jobs:

build:

runs-on: [ ubuntu-latest ] steps:

- name: Checkout Source

uses: actions/checkout@v3

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 37

- name: Polaris Scan

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0) with:

polaris\_serverUrl: *${{ secrets.POLARIS\_SERVERURL }}* polaris\_accessToken: *${{ secrets.POLARIS\_ACCESSTOKEN }}* polaris\_application\_name: *${{ github.event.repository.name }}* polaris\_project\_name: *${{ github.event.repository.name }}* ### Accepts Multiple Values

polaris\_assessment\_types: "SAST,SCA"

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # include\_diagnostics: true

##### Table 1. List of mandatory and optional parameters for Polaris

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
| polaris\_serverUrl | Polaris URL | Mandatory |
| polaris\_accessToken | Polaris Access token | Mandatory |
| polaris\_application\_name | Polaris Application name | Mandatory |
| polaris\_project\_name | Polaris Project name | Mandatory |

polaris\_assessment\_types

Polaris assessment types. Exam­ ple: SCA or SAST or SAST,SCA

Mandatory

## Using Synopsys GitHub Action for Black Duck

The Synopsys Action supports both self-hosted (e.g. on-prem) and Synopsys-hosted Black Duck Hub instances.

In the default Black Duck Hub permission model, projects and project versions are created on the fly and as needed. Ensure that permissions needed to create projects and project versions are granted on Black Duck Hub.

Synopsys action requires that you run full “intelligent” Black Duck scans for SCM push events and “rapid” ephemeral scans for SCM pull request events as shown in the example below.

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 38



**Note:**

Detect specific options can be passed to Synopsys Bridge thru Detect environment variables.

Below is an example of a *workflow*.yml file configured for Black Duck.

name: bd-sig-action

on:

push:

branches: [ main, master, develop, stage, release ] pull\_request:

branches: [ main, master, develop, stage, release ] workflow\_dispatch:

jobs:

build:

runs-on: [ ubuntu-latest ] steps:

* name: Checkout Source uses: actions/checkout@v3
* name: Black Duck Full Scan

if: *${{ github.event\_name != 'pull\_request' }}*

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0)

### Use below configuration to set specific detect environment variables env:

DETECT\_PROJECT\_NAME: *${{ github.event.repository.name }}*

with:

blackduck\_url: *${{ secrets.BLACKDUCK\_URL }}* blackduck\_apiToken: *${{ secrets.BLACKDUCK\_API\_TOKEN }}* blackduck\_scan\_full: true

### Accepts Multiple Values blackduck\_scan\_failure\_severities: 'BLOCKER,CRITICAL'

### Uncomment below configuration to enable automatic fix pull request creation if vulnerabilities are

reported

# blackduck\_automation\_fixpr: true

# github\_token: *${{ secrets.GITHUB\_TOKEN }}* # Mandatory when blackduck\_automation\_fixpr is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 39

# include\_diagnostics: true

- name: Black Duck PR Scan

if: *${{ github.event\_name == 'pull\_request' }}*

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0)

### Use below configuration to set specific detect environment variables env:

DETECT\_PROJECT\_NAME: *${{ github.event.repository.name }}*

with:

blackduck\_url: *${{ secrets.BLACKDUCK\_URL }}* blackduck\_apiToken: *${{ secrets.BLACKDUCK\_API\_TOKEN }}* blackduck\_scan\_full: false

### Below configuration is used to enable automatic pull request comment based on Black Duck scan result blackduck\_automation\_prcomment: true

github\_token: *${{ secrets.GITHUB\_TOKEN }}* # Mandatory when blackduck\_automation\_prcomment is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

**Creating Fix Pull Requests**

* **blackduck\_automation\_fixpr**: By default, fix pull request creation is disabled (Synopsys Action will not create fix pull requests for vulnerable direct dependencies.). To enable this feature, set blackduck\_automation\_fixpr as true.
* **github\_token**: You must pass github\_token parameter with required permissions. The token can be GitHub secrets.GITHUB\_TOKEN with required permissions. For more information on GitHub tokens see the [GitHub documentation](https://docs.github.com/en/actions/security-guides/automatic-token-authentication)
* Due to rate limit restriction of GitHub rest API calls, note that GitHub might limit the number of pull requests that are created by Synopsys Action.

##### Table 2. List of mandatory and optional parameters for Black Duck

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
| blackduck\_url | Black Duck URL | Mandatory |
| blackduck\_apiToken | Black Duck API token | Mandatory |
| blackduck\_install\_directory | Installation directory for Black Duck | Optional |

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 40

##### Table 2. List of mandatory and optional parameters for Black Duck (continued)

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
| blackduck\_scan\_full | Specifies whether full scan is re­ quired or not.  Must be set to true for push events and false for pull request events.  Default: false | Optional |
| blackduck\_scan\_failure\_severi• ties | Black Duck scan failure severi­ ties.  Supported values: ALL, NONE, BLOCKER, CRITICAL, MA­ JOR, MINOR, OK, TRIVIAL, UNSPECIFIED | Optional |
| blackduck\_automation\_prcomment | Option to enable automatic cre­ ation pull request comments for new issues found in the pull re­ quest.  Baseline full scan results must exist on the server for this feature to work.  Default: false | Optional |
| blackduck\_automation\_fixpr | Option to enable automatic creation for fix pull requests for vulnerable direct dependencies.  Default: false | Optional |

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 41

##### Table 2. List of mandatory and optional parameters for Black Duck (continued)

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
|  | Black Duck automation fix pull request is currently supported for NPM projects only. |  |
| github\_token | GitHub Access Token  Example: github\_token: ${{ se• crets.GITHUB\_TOKEN }} | Mandatory if blackduck\_automa• tion\_fixpr or blackduck\_automa• tion\_prcomment is set as true |

## Using Synopsys GitHub Action for Coverity Cloud Deployment with Thin Client

Synopsys GitHub Action only supports the Kubernetes-based Coverity cloud deployment model.

On push events, a full Coverity scan will be run and results are committed to the Coverity Connect database.

On pull request events, comments are added to pull requests for new issues found by the scan if coverity\_automation\_prcomment is set to true (see example below). Note that scan results are not committed to Coverity Connect database in this case.

Before running the pipeline with Synopsys Action, make sure the specified project and stream exist in your Coverity Connect server environment.

Below is an example of a *workflow*.yml file configured for Coverity Cloud Deployment.

name: cnc-sig-action on:

push:

branches: [ main, master, develop, stage, release ] pull\_request:

branches: [ main, master, develop, stage, release ] workflow\_dispatch:

jobs:

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 42

build:

runs-on: [ ubuntu-latest ]

steps:

* name: Checkout Source

uses: actions/checkout@v3

* name: Coverity Full Scan

if: *${{ github.event\_name != 'pull\_request' }}*

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0)

with:

coverity\_url: *${{ secrets.COVERITY\_URL }}*

coverity\_user: *${{ secrets.COVERITY\_USER }}*

coverity\_passphrase: *${{ secrets.COVERITY\_PASSPHRASE }}*

coverity\_project\_name: *${{ github.event.repository.name }}*

coverity\_stream\_name: *${{ github.event.repository.name }}*-*${{ github.ref\_name }}*

coverity\_policy\_view: 'Outstanding Issues'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

* name: Coverity PR Scan

if: *${{ github.event\_name == 'pull\_request' }}*

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0)

with:

coverity\_url: *${{ secrets.COVERITY\_URL }}*

coverity\_user: *${{ secrets.COVERITY\_USER }}*

coverity\_passphrase: *${{ secrets.COVERITY\_PASSPHRASE }}*

coverity\_project\_name: *${{ github.event.repository.name }}*

coverity\_stream\_name: *${{ github.event.repository.name }}*-*${{ github.base\_ref }}*

### Below configuration is used to enable feedback from Coverity security testing as pull request comment

coverity\_automation\_prcomment: true

github\_token: *${{ secrets.GITHUB\_TOKEN }}* # Mandatory when coverity\_automation\_prcomment is set to 'true'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 43

##### Table 3. List of mandatory and optional parameters for Coverity

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
| coverity\_url | Coverity URL | Mandatory |
| coverity\_user | Coverity username | Mandatory |
| coverity\_passphrase | Coverity passphrase | Mandatory |
| coverity\_project\_name | Coverity project name.  Tip: Many customers prefer to set their Coverity project and stream names to match the GitHub repository name | Mandatory |
| coverity\_stream\_name | Coverity stream name | Mandatory |
| coverity\_install\_directory | Installation directory of Coverity | Optional |
| coverity\_policy\_view | ID or name of policy view to be used to enforce the “break the build” policy.  If issues are found in the speci­ fied this view, build will be failed.  Example: coverity\_policy\_view: '100001' or coverity\_policy\_• view: 'Outstanding Issues' | Optional |
| coverity\_automation\_prcomment | Option to enable automatic cre­ ation pull request comments for new issues found in the pull re­ quest.  Baseline full scan results must exist on the server for this feature to work.  Default: false | Optional |

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 44

##### Table 3. List of mandatory and optional parameters for Coverity (continued)

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
|  |  |  |
| github\_token | GitHub Access Token  Example: github\_token: ${{ se• crets.GITHUB\_TOKEN }} | Mandatory if coverity\_automa­ tion\_prcomment is set as true |

## Additional GitHub Configuration

The following parameters can be used for Polaris, Black Duck or Coverity Connect.

* synopsys\_bridge\_path: Provides the path to Synopsys Bridge.



**Note:**

If this is not explicitly specified, then the integration defaults to $HOME/synopsys-bridge. If the installed version of Synopsys Bridge is not the latest, then the latest version of Synopsys Bridge is downloaded unless you specify the version to use explicitly (as documented below).

* bridge\_download\_url: Specifies the URL to the Synopsys Bridge zip file to be downloaded and used.



**Note:**

If bridge\_download\_url is not provided, Synopsys GitHub Action downloads the latest version of Synopsys Bridge from the default SIG-REPO download location.

* bridge\_download\_version: Specifies the Synopsys Bridge version to use. If provided, the specified version of Synopsys Bridge will be automatically downloaded and used. If not, the latest version is downloaded and used.
* include\_diagnostics: When set to true, Synopsys Bridge diagnostic files are created and posted to GitHub. Additionally, diagnostics\_retention\_days can be used to specify the number of days the diagnostics files are retained for. Default value is 90. Accepted range of values is from 1 to 90.

# Chapter 5. GitLab – Synopsys Template

Synopsys GitLab Template can be used to integrate Synopsys security testing into your CI pipeline.

You can download Synopsys GitLab Template using the GitLab Marketplace link: <https://gitlab.com/synopsys/synopsys-template>.

#### Additional information

For additional GitLab integration information, see:

* [GitLab Prerequisites *(on*](#_bookmark34)[*page 45)*](#_bookmark34)
* GitLab Runner Setup *(on page )*
* [Using Synopsys GitLab Template with Polaris *(on*](#_bookmark35)[*page 46)*](#_bookmark35)
* [Using the Synopsys GitLab Template with Black Duck *(on*](#_bookmark36)[*page 48)*](#_bookmark36)
* [Using the Synopsys GitLab Template for Coverity Cloud Deployment with Thin Client *(on*](#_bookmark37)[*page 51)*](#_bookmark37)
* [Additional GitLab Configuration *(on*](#_bookmark38)[*page 54)*](#_bookmark38)

## GitLab Prerequisites

Before configuring Synopsys Template into your GitLab pipeline, you must meet the following prerequisites:

#### GitLab Runner Setup

* GitLab Runner is an application that works with GitLab CI/CD to run jobs in a pipeline. To use GitLab Runner in your project, you must have the maintainer or owner role for the project.
* A GitLab runner can be self-managed or SaaS runners managed by GitLab.
* A GitLab self-managed runner can be installed and used on GNU/Linux, macOS and Windows. For more details refer: [Install GitLab Runner](https://docs.gitlab.com/runner/install/)
* To set up project specific self-managed runner, go to (**Project Settings** → **CI/CD** → **Runners**) and

configure.

* During runner registration, choose executor as shell.
* Make sure you have curl and unzip package tools installed in self-managed/SaaS runner (Linux/ Mac).
* Synopsys Template supports both Project runners and Shared runners (except Shared Mac Runners).

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 46

#### Configure GitLab Variables

* Sensitive data such as access tokens, user names, passwords and even URLs must be configured using GitLab variables.
* These can be added at the Project, Group or Global scopes (Global for self-managed GitLab instances only).
* To add variables, go to **Settings** → **CI/CD** → **Variables**. Be sure to mask passwords and tokens to

avoid them being exposed in logs. For more details see [GitLab CI/CD variables](https://docs.gitlab.com/ee/ci/variables).

#### Configure Gitlab User Token

* BRIDGE\_GITLAB\_USER\_TOKEN is required as input when running Black Duck Fix PR, Black Duck/ Coverity PR Comment.
* Generate a Personal Access Token (PAT) from GitLab (**User Settings** → **Access Tokens**) and store

it as secret variable or store and fetch it from vault.

* PAT must have **api** scope to perform Black Duck Fix PR or Black Duck/Coverity PR Comment. For more details, see: [Personal access tokens](https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html)

#### Create a .gitlab-ci.yml file

* Before running a pipeline using the Synopsys Template, add a .gitlab-ci.yml file to your project by adding an include entry.
* Push those changes and a GitLab runner picks up the job and initiates the pipeline.

## Using the Synopsys GitLab Template with Polaris

Before running a pipeline using the Synopsys GitLab Template with Polaris, you must set the appropriate applications and entitlements in your Polaris environment.

Using Synopsys Template, you can perform scans on push events to main branches. Pull request scanning is currently not supported for Polaris.

It is recommended that you configure sensitive information such as access tokens and URLs using GitLab secrets.

To use Synopsys Template, add .gitlab-ci.yml to your project by using an

Include entry, as shown in the example below.

include:

- project: synopsys/synopsys-template ref: v1.1.0

file: templates/synopsys-template.yml

### Use below configuration for accessing synopsys-template in GitLab self-managed

# - remote: 'https://gitlab.com/synopsys/synopsys-template/-/raw/v1.0.0/templates/synopsys-template.yml' variables:

BRIDGE\_POLARIS\_SERVERURL: *$POLARIS\_SERVER\_URL*

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 47

BRIDGE\_POLARIS\_ACCESSTOKEN: *$POLARIS\_ACCESS\_TOKEN* BRIDGE\_POLARIS\_APPLICATION\_NAME: *$CI\_PROJECT\_NAME* BRIDGE\_POLARIS\_PROJECT\_NAME: *$CI\_PROJECT\_NAME*

### Accepts Multiple Values BRIDGE\_POLARIS\_ASSESSMENT\_TYPES: 'SCA,SAST'

stages:

- polaris\_scan synopsys\_template\_execution:

stage: polaris\_scan tags:

- linux # Name of your GitLab runner extends: .run-synopsys-tools # Used for bash

# extends: .run-synopsys-tools-powershell # Used for powershell

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # variables:

# INCLUDE\_DIAGNOSTICS: 'true'

# artifacts:

# when: always # paths:

# - .bridge

Once you push the changes above, an active runner will pick up the job and initiate the pipeline.

##### Table 4. List of mandatory and optional parameters for Polaris

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
| BRIDGE\_POLARIS\_SERVERURL | Polaris server URL | Mandatory |
| BRIDGE\_POLARIS\_ACCESSTOKEN | Polaris access token | Mandatory |
| BRIDGE\_POLARIS\_APPLICATION\_• NAME | Application name in Polaris | Mandatory |
| BRIDGE\_POLARIS\_PROJECT\_NAME | Project name in Polaris | Mandatory |

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 48

##### Table 4. List of mandatory and optional parameters for Polaris (continued)

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory / Optional** |
| BRIDGE\_POLARIS\_ASSESSMENT\_• TYPES | Polaris assessment types Accepted values: SCA or SAST or SAST,SCA | Mandatory |

## Using the Synopsys GitLab Template with Black Duck

Synopsys GitLab Template supports both self-hosted (on-prem) and Synopsys-hosted Black Duck Hub instances.

In the default Black Duck Hub permission model, projects and project versions are created on the fly as needed. Ensure that permissions needed to create projects and project versions are granted on Black Duck Hub.

Synopsys Template requires that you run full “intelligent” Black Duck scans for SCM push events and “rapid” ephemeral scans for SCM pull request events as shown in the example below.



**Note:**

Detect specific options can be passed to Synopsys Bridge thru Detect environment variables.

To use Synopsys GitLab Template with Black Duck, add .gitlab-ci.yml to your project using an include entry as shown the example below.

include:

- project: synopsys/synopsys-template ref: v1.1.0

file: templates/synopsys-template.yml

### Use below configuration for accessing synopsys-template in Gitlab self-managed

# - remote: 'https://gitlab.com/synopsys/synopsys-template/-/raw/v1.1.0/templates/synopsys-template.yml'

stages:

- blackduck\_scan

variables:

SCAN\_BRANCHES: "/^(main|master|develop|stage|release|feature\_branch)$/" # Add branches where you want to run Black Duck scan

synopsys\_template\_execution: stage: blackduck\_scan variables:

BRIDGE\_BLACKDUCK\_URL: *$BLACKDUCK\_URL*

BRIDGE\_BLACKDUCK\_TOKEN: *$BLACKDUCK\_API\_TOKEN*

### Use below configuration to set specific detect environment variables

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 49

DETECT\_PROJECT\_NAME: *$CI\_PROJECT\_NAME*

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # INCLUDE\_DIAGNOSTICS: 'true'

# artifacts:

# when: always # paths:

# - .bridge rules:

### Use below configuration to run Black Duck full scan

- if: (*$CI\_COMMIT\_BRANCH* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* != 'merge\_request\_event') variables:

BRIDGE\_BLACKDUCK\_SCAN\_FULL: 'true'

### Accepts Multiple Values BRIDGE\_BLACKDUCK\_SCAN\_FAILURE\_SEVERITIES: 'BLOCKER,CRITICAL'

### Uncomment below configuration to enable automatic fix pull request creation if vulnerabilities are reported # BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR: 'true'

BRIDGE\_GITLAB\_USER\_TOKEN: *$GITLAB\_USER\_TOKEN* # Mandatory when BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR is set to

'true'

### Use below configuration to run Black Duck PR scan

- if: (*$CI\_MERGE\_REQUEST\_TARGET\_BRANCH\_NAME* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* == 'merge\_request\_event') variables:

BRIDGE\_BLACKDUCK\_SCAN\_FULL: 'false' BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT: 'true'

BRIDGE\_GITLAB\_USER\_TOKEN: $GITLAB\_USER\_TOKEN

tags:

- linux # Name of your Gitlab runner extends: .run-synopsys-tools # Used for bash.

#extends: .run-synopsys-tools-powershell # Used for powershell

##### Table 5. List of mandatory and optional parameters for Black Duck

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory/Optional** |
| BRIDGE\_BLACKDUCK\_URL | Black Duck server URL | Mandatory |
| BRIDGE\_BLACKDUCK\_TOKEN | Black Duck API token | Mandatory |
| BRIDGE\_BLACKDUCK\_INSTALL\_• DIRECTORY | Installation directory for Black Duck | Optional |

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 50

##### Table 5. List of mandatory and optional parameters for Black Duck (continued)

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory/Optional** |
| BRIDGE\_BLACKDUCK\_SCAN\_FULL | Specifies whether full scan is re­ quired or not.  Must be set to true for push events and false for pull request events.  Default: false | Optional |
| BRIDGE\_BLACKDUCK\_SCAN\_• FAILURE\_SEVERITIES | Black Duck scan failure severi­ ties used to decide if build should be broken.  Supported values: ALL, NONE, BLOCKER, CRITICAL, MA­ JOR, MINOR, OK, TRIVIAL, UNSPECIFIED | Optional |
| BRIDGE\_BLACKDUCK\_AUTOMATION\_• FIXPR | Option to enable automatic creation for fix pull requests for vulnerable direct dependencies.  Default: false  Note: Black Duck automation fix pull request is currently supported for NPM projects only. | Optional |
| BRIDGE\_BLACKDUCK\_AUTOMATION\_• PRCOMMENT | Option to enable automatic cre­ ation pull request comments for new issues found in the pull re­ quest.  Baseline full scan results must exist on the server for this feature to work.  Default: false | Optional |

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 51

##### Table 5. List of mandatory and optional parameters for Black Duck (continued)

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory/Optional** |
| BRIDGE\_GITLAB\_USER\_TOKEN | Gitlab User Access Token  Example: BRIDGE\_GITLAB\_USER\_TO• KEN: $GITLAB\_ACCESS\_TOKEN | Mandatory when BRIDGE\_BLACK• DUCK\_AUTOMATION\_PRCOMMENT or BRIDGE\_BLACKDUCK\_AUTOMATION\_•  FIXPR is set as true. |

## Using the Synopsys GitLab Template for Coverity Cloud Deployment with Thin Client

Synopsys Template only supports the Kubernetes-based Coverity cloud deployment model.

On push events, a full Coverity scan will be run and results are committed to the Coverity server database.

On pull request events, comments are added to pull requests for new issues found by the scan if BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT is set to true (see example below). Note that scan results are not committed to Coverity server database in this case.

Before running the pipeline with Synopsys Template, make sure the specified project and stream exist in your Coverity server.

To use Synopsys GitLab Template with Coverity, add .gitlab-ci.yml to your project using an include entry as shown the example below

include:

- project: synopsys/synopsys-template ref: v1.1.0

file: templates/synopsys-template.yml

### Use below configuration for accessing synopsys-template in Gitlab self-managed

# - remote: 'https://gitlab.com/synopsys/synopsys-template/-/raw/v1.1.0/templates/synopsys-template.yml'

stages:

- coverity\_scan

variables:

SCAN\_BRANCHES: "/^(main|master|develop|stage|release|feature\_branch)$/" # Add branches where you want to run Coverity scan

synopsys\_template\_execution: stage: coverity\_scan

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 52

variables:

BRIDGE\_COVERITY\_CONNECT\_URL: *$COVERITY\_URL* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME: *$COVERITY\_USER* BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD: *$COVERITY\_PASSWORD* BRIDGE\_COVERITY\_CONNECT\_PROJECT\_NAME: *$CI\_PROJECT\_NAME*

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # INCLUDE\_DIAGNOSTICS: 'true'

# artifacts:

# when: always # paths:

# - .bridge rules:

* if: (*$CI\_COMMIT\_BRANCH* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* != 'merge\_request\_event') variables:

BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$CI\_PROJECT\_NAME*-*$CI\_COMMIT\_BRANCH*

BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: 'Outstanding Issues' ### Use below configuration to run Coverity PR scan

* if: (*$CI\_MERGE\_REQUEST\_TARGET\_BRANCH\_NAME* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* == 'merge\_request\_event') variables:

BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$CI\_PROJECT\_NAME*-*$CI\_MERGE\_REQUEST\_TARGET\_BRANCH\_NAME*

### Below configuration is used to enable feedback from Coverity security testing as pull request comment BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT: 'true'

BRIDGE\_GITLAB\_USER\_TOKEN: *$GITLAB\_USER\_TOKEN* # Mandatory when BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT is set to

'true' tags:

* linux # Name of your Gitlab runner extends: .run-synopsys-tools # Used for bash.

#extends: .run-synopsys-tools-powershell # Used for powershell

##### Table 6. List of mandatory and optional parameters for Coverity cloud

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory/Optional** |
| BRIDGE\_COVERITY\_CONNECT\_URL | Coverity server URL | Mandatory |
| BRIDGE\_COVERITY\_CONNECT\_USER\_• NAME | Coverity username | Mandatory |
| BRIDGE\_COVERITY\_CONNECT\_USER\_• PASSWORD | Coverity passphrase | Mandatory |

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 53

##### Table 6. List of mandatory and optional parameters for Coverity cloud (continued)

|  |  |  |
| --- | --- | --- |
| **Input Parameter** | **Description** | **Mandatory/Optional** |
| BRIDGE\_COVERITY\_CONNECT\_• PROJECT\_NAME | Project name in Coverity | Mandatory |
| BRIDGE\_COVERITY\_CONNECT\_• STREAM\_NAME | Stream name in Coverity | Mandatory |
| BRIDGE\_COVERITY\_INSTALL\_DIREC• TORY | Installation directory of Coverity | Optional |
| BRIDGE\_COVERITY\_CONNECT\_POLI• CY\_VIEW | ID or name of policy view to be used to enforce the “break the build” policy.  If issues are found in the speci­ fied this view, build will be failed.  Example: coverity\_policy\_view: '100001' or coverity\_policy\_• view: 'Outstanding Issues' | Optional |
| BRIDGE\_COVERITY\_AUTOMATION\_• PRCOMMENT | Option to enable automatic cre­ ation pull request comments for new issues found in the pull re­ quest.  Merge Request must be created first from feature branch to main branch to run Coverity PR Com­ ment.  Default: false | Optional |
| BRIDGE\_GITLAB\_USER\_TOKEN | Gitlab User Access Token  Example: BRIDGE\_GITLAB\_USER\_TO• KEN: $GITLAB\_USER\_TOKEN | Mandatory when BRIDGE\_COVERI• TY\_AUTOMATION\_PRCOMMENT is set  as true. |

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 54

## Additional GitLab Configuration

The following optional parameters can be used for Polaris, Black Duck or Coverity Connect.

* SYNOPSYS\_BRIDGE\_PATH: Use this to specify the path to Synopsys Bridge . Optional.



**Note:**

If this is not explicitly specified, then the integration defaults to $HOME/synopsys-bridge. If the installed version of Synopsys Bridge is not the latest, then the latest version of Synopsys Bridge is downloaded unless you specify the version to use explicitly (as documented below).

* DOWNLOAD\_BRIDGE\_URL: Use this to specify the URL to the Synopsys Bridge zip file to be downloaded from and used.



**Note:**

If DOWNLOAD\_BRIDGE\_URL is not provided, Synopsys GitLab Temp,late downloads the latest version of Synopsys Bridge from the [Synopsys Artifactory](https://sig-repo.synopsys.com/artifactory/bds-integrations-release/com/synopsys/integration/synopsys-bridge/).

* DOWNLOAD\_BRIDGE\_VERSION: Use this to specify the Synopsys Bridge version to use. If provided, the specified version of Synopsys Bridge will be automatically downloaded and used. If not, the latest version is downloaded and used.
* INCLUDE\_DIAGNOSTICS: When set to true, Synopsys Bridge diagnostic files are created.



**Note:**

While including Synopsys Bridge diagnostic files, default expiry time for uploaded artifacts is 30 days. Refer to SCM documentation for more details :[https://docs.gitlab.com/ee/ci/](https://docs.gitlab.com/ee/ci/jobs/job_artifacts.html) [jobs/job\_artifacts.html](https://docs.gitlab.com/ee/ci/jobs/job_artifacts.html).

# Chapter 6. Azure DevOps - Synopsys Security Scan

Synopsys Security Scan Extension for Azure DevOps enables you to integrate Synopsys security testing into your Azure pipeline.

You can download Synopsys Security Scan for Azure DevOps using the Azure DevOps marketplace link: [https://](https://marketplace.visualstudio.com/items?itemName=synopsys-security-scan.synopsys-security-scan) [marketplace.visualstudio.com/items?itemName=synopsys-security-scan.synopsys-security-scan](https://marketplace.visualstudio.com/items?itemName=synopsys-security-scan.synopsys-security-scan).

#### Additional Info

For additional Azure integration information, see:

* [Azure Prerequisites *(on*](#_bookmark40)[*page 55)*](#_bookmark40)
* [Using Azure DevOps Extension with Polaris *(on*](#_bookmark41)[*page 56)*](#_bookmark41)
* [Using Azure DevOps Extension with Black Duck *(on*](#_bookmark42)[*page 57)*](#_bookmark42)
* [Using Azure DevOps Extension with Coverity Connect with Thin Client *(on*](#_bookmark43)[*page 60)*](#_bookmark43)
* [Additional Azure Configuration *(on*](#_bookmark38)[*page 54)*](#_bookmark38)

## Azure Prerequisites

Before adding Synopsys Security Scan in your azure pipeline, you must meet the following prerequisites :

#### Azure Agent Setup

Azure agents are required and can be installed and used on GNU/Linux, macOS, Windows and Docker. See [https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser)

[devops&tabs=browser](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser) for details. You can use Microsoft-hosted agents as well to scan your code using Azure Pipelines.

#### Configure Variables

Sensitive data such as access tokens, user names, passwords and even URLs must be configured using variable groups (**Project** → **Pipelines** → **Library** → **New Variable Group**).

AZURE\_TOKEN is required as input when running Black Duck Fix PR, Black Duck/Coverity PR Comment. There are two different types of tokens to pass to AZURE\_TOKEN:

* To use AZURE\_TOKEN: $(System.AccessToken), you must enable this in the Azure interface. Go to **Project** → **Project Settings** → **Repository** → **Security** → **Build Service** and set **Contribute to pull requests**, **Create branch** and **Delete or disable repository** to **Allow**. Confirm System.AccessToken

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 56

has **Contribute to PR** permissions (**Project** → **Project Settings** → **Repositories** → **Security** → **Build Service User**).

* To use AZURE\_TOKEN: $(PAT\_TOKEN), PAT token should have minimum permissions **Code - Full** and

**Pull Request Threads - Read & write**. See [Use personal access tokens](https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=Windows) for more details.

If you like Synopsys Security Scan to add comments to pull requests (supported forBlack Duck and Coverity), enable **Buildvalidation policy** (**Project**→**ProjectSettings** → **Repositories**→**Branch Policy** →**Add branch protection**) to trigger the pipeline on any PR or push event to a branch (usually main or master branch). See [Build Validation](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops&tabs=browser&build-validation) for more details.

#### Configure Azure Pipeline

Create a new pipeline or use existing pipeline (**Project** → **Pipelines** → **New Pipeline**) and configure required fields. Push those changes and agent will pick up the job and initiate the pipeline.

## Using Azure DevOps Extension with Polaris

Before running a pipeline using the Synopsys Security Scan with Polaris, you must set the appropriate applications and entitlements in your Polaris environment.

Using Synopsys Security Scan, you can perform scans on push events to main branches. Pull request scanning is currently not supported for Polaris.

To use Synopsys Security Scan:

1. Configure sensitive data such as usernames, passwords and URLs using pipeline variables.
2. Add azure-pipelines.yml to your project.
3. Push the changes and an agent will pick up the job and initiate the pipeline.

Here is an example azure- pipelines.yml that you can use with Polaris:

trigger:

- main

pool:

vmImage: ubuntu-latest

variables:

- group: polaris

steps:

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0) displayName: 'Polaris Scan' inputs:

BRIDGE\_POLARIS\_SERVERURL: *$(POLARIS\_SERVER\_URL)*

BRIDGE\_POLARIS\_ACCESSTOKEN: *$(POLARIS\_ACCESS\_TOKEN)*

BRIDGE\_POLARIS\_APPLICATION\_NAME: *$(Build.Repository.Name)*

BRIDGE\_POLARIS\_PROJECT\_NAME: *$(Build.Repository.Name)*

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 57

### Accepts Multiple Values BRIDGE\_POLARIS\_ASSESSMENT\_TYPES: 'SCA,SAST'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# INCLUDE\_DIAGNOSTICS: 'true'

##### Table 7. List of mandatory and optional parameters for Polaris below: Input Parameter Description

**Mandato­ ry / Optional**

BRIDGE\_POLARIS\_SERVERURL Polaris URL Mandatory

BRIDGE\_POLARIS\_ACCESSTOKEN Polaris access token Mandatory

BRIDGE\_POLARIS\_APPLICATION\_• NAME

Polaris Application name Mandatory

BRIDGE\_POLARIS\_PROJECT\_NAME Polaris Project name Mandatory

BRIDGE\_POLARIS\_ASSESSMENT\_• TYPES

Polaris assessment types.

Accepted values: SCA or SAST or SAST,SCA

Mandatory

## Using Azure DevOps Extension with Black Duck

Synopsys Security Scan supports both self-hosted (e.g. on-prem) and Synopsys-hosted Black Duck Hub instances.

In the default Black Duck Hub permission model, projects and project versions are created on the fly and as needed. Ensure that permissions needed to create projects and project versions are granted on Black Duck Hub.

Configure sensitive data like usernames, passwords and URLs using pipeline variables.



**Note:**

Detect specific options can be passed to Synopsys Bridge thru Detect environment variables.

Here is an example azure-pipelines.yml that you can use with Black Duck:

trigger:

- main

pool:

vmImage: ubuntu-latest

variables:

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 58

- group: blackduck

steps:

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0)

displayName: 'Black Duck Full Scan'

condition: not(eq(variables['Build.Reason'], 'PullRequest'))

### Use below configuration to set specific detect environment variables

env:

DETECT\_PROJECT\_NAME: *$(Build.Repository.Name)*

inputs:

BRIDGE\_BLACKDUCK\_URL: *$(BLACKDUCK\_URL)*

BRIDGE\_BLACKDUCK\_TOKEN: $*(BLACKDUCK\_TOKEN)*

BRIDGE\_BLACKDUCK\_SCAN\_FULL: true

### Accepts Multiple Values

BRIDGE\_BLACKDUCK\_SCAN\_FAILURE\_SEVERITIES: 'BLOCKER,CRITICAL'

### Uncomment below configuration to enable automatic fix pull request creation if vulnerabilities are reported

# BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR: true

# AZURE\_TOKEN: *$(System.AccessToken)* # Mandatory when BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR is set to 'true'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# INCLUDE\_DIAGNOSTICS: true

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0)

displayName: 'Black Duck PR Scan'

condition: eq(variables['Build.Reason'], 'PullRequest')

### Use below configuration to set specific detect environment variables

env:

DETECT\_PROJECT\_NAME: $*(Build.Repository.Name)*

inputs:

BRIDGE\_BLACKDUCK\_URL: *$(BLACKDUCK\_URL)*

BRIDGE\_BLACKDUCK\_TOKEN: *$(BLACKDUCK\_API\_TOKEN)*

BRIDGE\_BLACKDUCK\_SCAN\_FULL: false

### Below configuration is used to enable automatic pull request comment based on Black Duck scan result

BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT: true

AZURE\_TOKEN: *$(System.AccessToken)* # Mandatory when BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT is set to 'true'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# INCLUDE\_DIAGNOSTICS: true

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 59

##### Table 8. List of mandatory and optional parameters for Black Duck below:

**Input Parameter Description Mandatory / Optional**

BRIDGE\_BLACK• DUCK\_URL

BRIDGE\_BLACK• DUCK\_TOKEN

BRIDGE\_BLACK• DUCK\_INSTALL\_• DIRECTORY

BRIDGE\_BLACK• DUCK\_SCAN\_FULL

Black Duck URL Mandatory

Black Duck API token Mandatory

Installation directory for Black Duck Optional

Optional

Specifies whether full scan is re­ quired or not.

Must be set to true for push events and false for pull request events.

Default: false

BRIDGE\_BLACK• DUCK\_SCAN\_• FAILURE\_SEV• ERITIES

Black Duck scan failure severi­ ties used to decide if build should be broken.

Supported values: ALL, NONE, BLOCKER, CRITICAL, MAJOR, MINOR, OK, TRIVIAL, UNSPECIFIED

Optional

BRIDGE\_• BLACKDUCK\_AU• TOMATION\_PRCOM• MENT

Option to enable automatic cre­ ation pull request comments for new issues found in the pull re­ quest.

Baseline full scan results must exist on the server for this feature to work.

Default: false

**Note** - Feature is supported only through yaml configuration

Optional

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 60

##### Table 8. List of mandatory and optional parameters for Black Duck below: (continued)

**Input Parameter Description Mandatory / Optional**

BRIDGE\_BRIDGE\_• BLACKDUCK\_AU• TOMATION\_FIXPR

Option to enable automatic creation for fix pull requests for vulnerable direct de­ pendencies.

Default: false

##### Note - Black Duck automation fix pull request is currently supported for npm projects only.

**Note** - Feature is supported only through yaml configuration

Optional

AZURE\_TOKEN

Azure Access Token

Example: AZURE\_TOKEN: $(System.Access• Token) or AZURE\_TOKEN: $(PAT\_TOKEN)

Mandatory if BRIDGE\_BLACKDUCK\_AU• TOMATION\_PRCOMMENTorBRIDGE\_BRIDGE\_BLACK•

DUCK\_AUTOMATION\_FIXPR is set true.

## Using Azure DevOps Extension with Coverity Connect with Thin Client

Synopsys Security Scan only supports the Coverity Connect with thin client deployment model.

On push events, a full Coverity scan will be run and results are committed to the Coverity server database.

On pull request events, comments are added to pull requests for new issues found by the scan if BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT is set to true (see example below). Note that scan results are not committed to Coverity server database in this case.

Before running the pipeline with Synopsys Security Scan, make sure the specified project and stream exist in your Coverity server. Configure sensitive data such as usernames and passwords using pipeline variables.

Here is an example azure-pipelines.yml that you can use to integrate Coverity into your Azure pipeline:

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 61

trigger:

* main

pool:

vmImage: ubuntu-latest

variables:

- group: coverity

steps:

* task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0)

displayName: 'Coverity Full Scan'

condition: not(eq(variables['Build.Reason'], 'PullRequest'))

inputs:

BRIDGE\_COVERITY\_CONNECT\_URL: *$(COVERITY\_URL)*

BRIDGE\_COVERITY\_CONNECT\_USER\_NAME: *$(COVERITY\_USER)*

BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD: *$(COVERITY\_PASSPHRASE)*

BRIDGE\_COVERITY\_CONNECT\_PROJECT\_NAME: *$(Build.Repository.Name)*

BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$(Build.Repository.Name)*-*$(Build.SourceBranchName)*

BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: 'Outstanding Issues'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

* task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0)

displayName: 'Coverity PR Scan'

condition: eq(variables['Build.Reason'], 'PullRequest')

inputs:

BRIDGE\_COVERITY\_CONNECT\_URL: *$(COVERITY\_URL)*

BRIDGE\_COVERITY\_CONNECT\_USER\_NAME: *$(COVERITY\_USER)*

BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD: *$(COVERITY\_PASSPHRASE)*

BRIDGE\_COVERITY\_CONNECT\_PROJECT\_NAME: *$(Build.Repository.Name)*

BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$(Build.Repository.Name)*-*$(Build.targetBranchName)*

### Below configuration is used to enable feedback from Coverity security testing as pull request comment

BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT: true

AZURE\_TOKEN: $(System.AccessToken) # Mandatory when BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT is set to 'true'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 62

##### Table 9. List of mandatory and optional parameters for Coverity below:

**Input Parameter Description Mandatory / Optional**

BRIDGE\_COVERITY\_• CONNECT\_URL

BRIDGE\_COVERITY\_• CONNECT\_USER\_NAME

BRIDGE\_COVERITY\_• CONNECT\_USER\_PASS• WORD

BRIDGE\_COVERITY\_• CONNECT\_PROJECT\_• NAME

BRIDGE\_COVERITY\_• CONNECT\_STREAM\_• NAME

BRIDGE\_COVERITY\_• INSTALL\_DIRECTORY

BRIDGE\_COVERITY\_• CONNECT\_POLICY\_• VIEW

Coverity URL Mandatory

Coverity Username Mandatory

Coverity Password Mandatory

Coverity Project Name Mandatory

Coverity Stream name Mandatory

Installation directory of Coverity Optional

Optional

ID or name of policy view to be used to enforce the

“break the build” policy.

If issues are found in the specified this view, build will be failed.

Example: coverity\_policy\_view: '100001' or

coverity\_policy\_view: 'Outstanding Issues'

BRIDGE\_COVERITY\_• AUTOMATION\_PRCOM• MENT

Option to enable automatic cre­ ation pull request comments for new issues found in the pull re­ quest.

Merge Request must be created first from feature branch to main branch to run Coverity PR Com­ ment.

Default: falseOptional

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 63

##### Table 9. List of mandatory and optional parameters for Coverity below: (continued)

**Input Parameter Description Mandatory / Optional**

Default: false

**Note** - Feature is supported only through yaml con­ figuration

AZURE\_TOKEN

Azure Access Token

Example: AZURE\_TOKEN: $(System.AccessToken) or

AZURE\_TOKEN: $(PAT\_TOKEN)

Mandatory if BRIDGE\_COVERI• TY\_AUTOMATION\_PRCOMMENT is set

true.

## Additional Azure Configuration

The following optional parameters can be used for Polaris, Black Duck or Coverity Connect.

* + SYNOPSYS\_BRIDGE\_PATH: Use this to specify the path to Synopsys Bridge. Optional.



**Note:**

If this is not explicitly specified, then the integration defaults to $HOME/synopsys-bridge. If the installed version of Synopsys Bridge is not the latest, then the latest version of Synopsys Bridge is downloaded unless you specify the version to use explicitly (as documented below).

* + BRIDGE\_DOWNLOAD\_URL: Use this to specify the URL to Synopsys Bridge zip file to be downloaded and used .



**Note:**

If bridge\_download\_url is not provided, Synopsys Security Scan will download the latest version of Synopsys Bridge from the [Synopsys Artifactory](https://sig-repo.synopsys.com/artifactory/bds-integrations-release/com/synopsys/integration/synopsys-bridge/).

* + BRIDGE\_DOWNLOAD\_VERSION: Use this to specify the Synopsys Bridge version to use. If provided, the specified version of Synopsys Bridge will be automatically downloaded and used. If not, the latest version is downloaded and used .

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 64



.

* + include\_diagnostics: When set to true, Synopsys Bridge diagnostic files are created. Azure DevOps no longer supports per-pipeline retention rules. The only way to configure retention policies for YAML and classic pipelines is through the project settings. For more details, see [Set run retention policies](https://learn.microsoft.com/en-us/azure/devops/pipelines/policies/retention?view=azure-devops&tabs=yaml&set-run-retention-policies).

# Chapter 7. Glossary

Here are terms and concepts used by Synopsys Bridge and the various Synopsys programs with which it interfaces.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Application** | The software security tool used to scan code. |
| **Application Security** | Application security is enhancing software features to functionality to prevent security threats. These include denial of service attacks, unauthorized data access, privilege escalation attacks, etc. Applica­ tion security is one of several levels of security used to protect sys­ tems. |
| **BDSA** | Black Duck Security Advisory, highly detailed open source vulnerabil­ ity records that are hand-crafted by the Synopsys Cybersecurity Re­ search Center (CyRC) |
| **Black Duck** | Software composition analysis (SCA) security scanning tool. Helps manage the security, quality, and license compliance risks of open source and third-party code in applications and containers. Bridge in­ tegrates with Black Duck. |
| **CI/CD** | Continuous Integration/Continuous Deployment, the process by which new checked-in code is automatically built, checked for securi­ ty issues, and packaged for deployment. |
| **CLI** | Command Line Interface |
| **Coverity** | Static analysis scanning tool (SAST), which scans source code for security flaws and coding standards compliance. Bridge does not in­ tegrate with Coverity, but does integrate with Coverity Connect and CNC. |
| **Coverity Connect** | A web-based platform for Coverity. Bridge supports Coverity Con­ nect. |
| **Coverity cloud deployment** | A cloud-native version of Coverity. Bridge supports Coverity cloud de­ ployment, and every place in this manual that references "Coverity Connect" also applies to Coverity cloud deployment. |

Synopsys Bridge CLI Guide | 7 - Glossary | 66

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **CVE** | Common Vulnerabilities and Exposures. A database of publicly iden­ tified, defined, and cataloged cybersecurity vulnerabilities. |
| **EULM** | End User License Management agreement |
| **GUI** | Graphic User Interface |
| **IAST** | Interactive application security testing (IAST) solutions help organi­ zations identify and manage security risks associated with vulnera­ bilities discovered in running web applications by continuously an­ alyzing all application interactions initiated by manual and/or auto­ mated tests to identify vulnerabilities in real time. |
| **Polaris** | Polaris is a cloud-native application security testing solution that pro­ vides both best-in-class SAST and SCA, making it easier to manage application security testing. Bridge integrates with Polaris. |
| **Rapid Scan Static (Sigma)** | Rapid Scan Static using the Sigma engine is a headless Static Appli­ cation Security Testing (SAST) scanner. |
| **RSQL** | REST Query Language |
| **Runner** | An application that runs a pipeline job from a CI/CD platform like GitHub or GitLab. |
| **SAST** | Static Analysis Security Testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabili­ ties. SAST scans an application before the code is compiled. Coveri­ ty is a SAST tool. |
| **SCA** | Software Composition Analysis (SCA) is an automated process iden­ tifying open source software in a codebase to evaluate security, li­ cense compliance, and code quality. Black Duck is an SCA tool. |
| **SCM** | Source Code Management. This usually refers to an online CI/CD SCM repo like GitHub, GitLab or Azure, all of which Synopsys offers integrations adaptors for. |