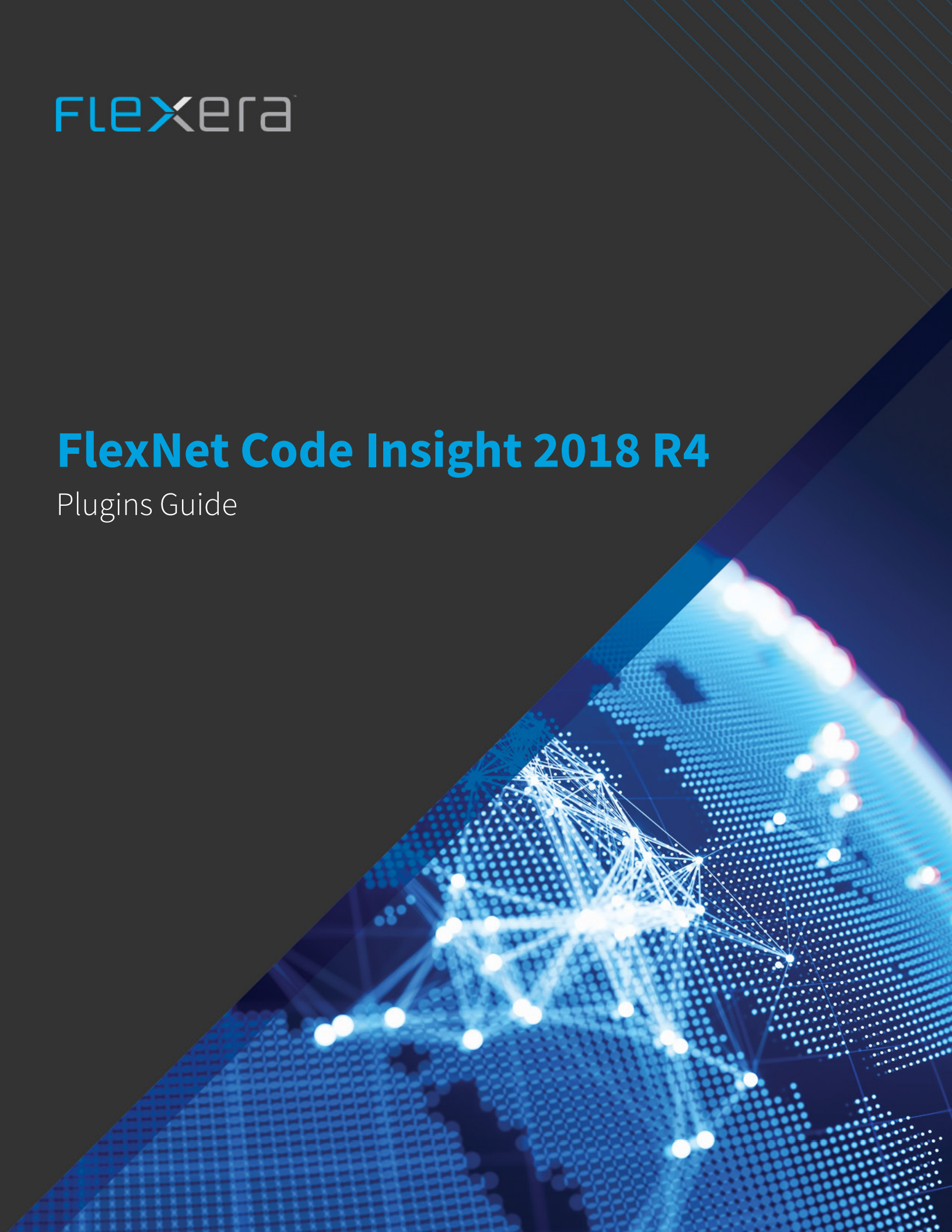




# FlexNet Code Insight 2018 R4

Plugins Guide



# Legal Information

**Book Name:** FlexNet Code Insight R4 Plugins Guide  
**Part Number:** FNCI-2018R4-PG00  
**Product Release Date:** January 2019

## Copyright Notice

Copyright © 2019 Flexera

This publication contains proprietary and confidential information and creative works owned by Flexera and its licensors, if any. Any use, copying, publication, distribution, display, modification, or transmission of such publication in whole or in part in any form or by any means without the prior express written permission of Flexera is strictly prohibited. Except where expressly provided by Flexera in writing, possession of this publication shall not be construed to confer any license or rights under any Flexera intellectual property rights, whether by estoppel, implication, or otherwise.

All copies of the technology and related information, if allowed by Flexera, must display this notice of copyright and ownership in full.

FlexNet Code Insight incorporates software developed by others and redistributed according to license agreements. Copyright notices and licenses for these external libraries are provided in a supplementary document that accompanies this one.

## Intellectual Property

For a list of trademarks and patents that are owned by Flexera, see <https://www.flexera.com/producer/company/about/intellectual-property/>. All other brand and product names mentioned in Flexera products, product documentation, and marketing materials are the trademarks and registered trademarks of their respective owners.

## Restricted Rights Legend

The Software is commercial computer software. If the user or licensee of the Software is an agency, department, or other entity of the United States Government, the use, duplication, reproduction, release, modification, disclosure, or transfer of the Software, or any related documentation of any kind, including technical data and manuals, is restricted by a license agreement or by the terms of this Agreement in accordance with Federal Acquisition Regulation 12.212 for civilian purposes and Defense Federal Acquisition Regulation Supplement 227.7202 for military purposes. The Software was developed fully at private expense. All other use is prohibited.

# Contents

<b>1</b>	<b>FlexNet Code Insight 2018 R4 Plugins Guide</b>	<b>7</b>
	About Scan Agent Plugins	7
	Contents of this Book	8
	Contacting Support	8
<b>2</b>	<b>Installing and Configuring Standard Plugins</b>	<b>9</b>
	About Scan Agent Plugins	10
	Overview of Available Plugins	10
	Preparing to Use the Plugins	11
	Generating a JWT Authorization Token	11
	Downloading Plugins	11
	Plugins for Integrated Development Environments (IDEs)	12
	Eclipse Plugin	12
	Prerequisites for the Eclipse Plugin	12
	Installing the Eclipse Plugin	13
	Configuring the Eclipse Plugin	15
	Running a Scan within Your Environment	16
	Uninstalling the Eclipse Plugin	17
	Plugins for Continuous Integration (CI) Tools	18
	Azure DevOps Extension	18
	Prerequisites for the Azure DevOps Extension	18
	Installing the Azure DevOps Extension	18
	Adding a Scan Task to Your Azure DevOps Agent Job	19
	Bamboo Plugin	21
	Prerequisites for the Bamboo Plugin	21
	Installing and Configuring the Bamboo Plugin	21
	GitLab Plugin	22
	Prerequisites for the GitLab Plugin	23

Installing the Generic Scan Agent on GitLab Runner .....	23
Configuring the CI/CD Pipeline to Run a Scan .....	23
Executing the Build .....	24
Jenkins Plugin .....	24
Prerequisites for the Jenkins Plugin .....	24
Setting Heap Size for the Jenkins Plugin .....	25
Setting Up the Code Insight Jenkins Plugin .....	25
Support for the Jenkins Pipeline .....	26
Providing the Pipeline Script for the Scan Step .....	27
Pipeline Code Examples for Running the Scan. ....	27
Scan Scheduler Plugin for Jenkins .....	28
TeamCity Plugin. ....	29
Prerequisites for TeamCity Plugin .....	30
Installing the Generic Scan Agent on the TeamCity Agent .....	30
Configuring a Build to Run a Code Insight Scan .....	30
Executing the Build .....	32
<b>Plugins for Package Managers and Build Tools .....</b>	<b>32</b>
Apache Ant Plugin .....	32
Prerequisites for the Apache Ant Plugin .....	32
Configuring the Apache Ant Plugin. ....	33
Executing the Scan. ....	33
Gradle Plugin .....	34
Prerequisites for the Gradle Plugin .....	34
Installing and Configuring the Gradle Plugin .....	35
Maven Plugin .....	36
More About the Maven Plugin .....	36
Prerequisites for the Maven Plugin. ....	36
Installing and Configuring the Maven Plugin .....	37
Cleaning the Application Project .....	39
Running the Maven Goal for the Scan .....	39
<b>Plugins for Binary Repositories .....</b>	<b>39</b>
JFrog Artifactory Plugin .....	39
Prerequisites for the Artifactory Plugin .....	40
Installing the Artifactory Plugin .....	40
Scanning an Artifactory Repository Using a Cron Job .....	40
Scanning an Artifactory Repository Using REST API. ....	41
Requirements When Using REST API to Scan Artifactory Repositories. ....	41
Scanning All Artifactory Repositories. ....	42
Scanning a Specific Artifactory Repository. ....	42
Reloading the Artifactory Plugin. ....	42
Scan Results. ....	42
<b>Plugins for Container Platforms .....</b>	<b>43</b>
Docker Images Plugin .....	43
Prerequisites for the Docker Images Plugin .....	43
Installing the Docker Images Plugin. ....	44
Launching the Docker Images Plugin .....	44

**Generic Scan Agent Plugin ..... 45**  
    Prerequisites for the Generic Scan Agent Plugin ..... 45  
    Running the Generic Scan Agent Plugin..... 46

**3   Developing Custom Plugins ..... 47**

**Scan Agent Framework ..... 47**  
        Features Provided by the Framework..... 48  
        Available Classes and Methods in the Framework..... 48  
        Property Settings..... 49

**Downloading the Scan Agent Toolkit ..... 50**

**Contents of the Scan Agent Toolkit ..... 50**

**Writing a Custom Scan Agent Plugin ..... 51**

**Deploying a Custom Scan Agent Plugin ..... 51**



# FlexNet Code Insight 2018 R4 Plugins Guide

FlexNet Code Insight empowers organizations to take control of and manage their use of open source software (OSS) and other third-party components. It helps development, legal, and security teams use automation to create a formal OSS strategy that balances business benefits and risk management.

The *FlexNet Code Insight Plugins Guide* describes how to install and configure a Code Insight standard scan agent plugin (or develop your own custom plugin) in your development environment to scan your codebases for OSS and third-party components.

This chapter covers the following information:

- [About Scan Agent Plugins](#)
- [Contents of this Book](#)
- [Contacting Support](#)

## About Scan Agent Plugins

FlexNet Code Insight supports a *scan agent plugin*, which is installed in your development environment to perform scans on a codebase as part of your software development process. This type of scan is an alternative to the scans performed by the Code Insight scan server on codebases that are uploaded to that server. (The process to upload a codebase to the scan server is described in *FlexNet Code Insight User Guide*.)

The scan agent plugin determines the appropriate codebase to be scanned within the context of an Engineering application (such as an IDE, source management, artifact repository, CI, build, testing, or installation application). The plugin then invokes a scan as part of the build process and sends the results back to Code Insight as published inventory awaiting review, management, and remediation. As with published inventory produced by the Code Insight scan server, published inventory produced by a scan agent plugin can be automatically reviewed by license or security policies during the scan and, for inventory not reviewed by policy, can be reviewed manually by legal or security experts. Security alerts with corresponding email notifications will be generated for any inventory item with new security vulnerabilities.

Code Insight offers a set of standard scan agent plugins that are pre-built and can be deployed immediately. It also provides a generic scan agent plugin (also pre-built) that can be used as a standalone scan agent to scan arbitrary file systems or that can be integrated with certain Engineering applications for automatic code scanning.

Additionally, Code Insight offers a Scan Agent toolkit that enables you to create a custom scan agent plugin that integrates with your development ecosystem.

## Contents of this Book

The *FlexNet Code Insight Plugins Guide* includes the following sections.

**Table 1-1 • FlexNet Code Insight Plugins Guide**

Chapter	Content
<b>Installing and Configuring Standard Plugins</b>	Describes how to install and configure the Code Insight standard scan agent plugins.
<b>Developing Custom Plugins</b>	Describes how to use the Scan Agent toolkit to develop custom scan agent plugins.

## Contacting Support

If you need further support, submit your questions through our online **Customer Community** portal:

<https://flexeracommunity.force.com/customer/>

If you do not have a login to the Customer Community, you can request one on the **Login Request** page of our site:

<https://flexeracommunity.force.com/customer/CCLoginRequest>

If you are unable to use the steps above, please visit the following site for other options to reach out to Flexera Support:

<https://flexeracommunity.force.com/customer/CCContactSupport>



# Installing and Configuring Standard Plugins

FlexNet Code Insight supports *scan agent plugins*, which are installed in development environments to perform scans on codebases as part of the software development process. These remote scans are alternatives to the scans performed by the Code Insight scan server on codebases that are uploaded to that server, as described in the *FlexNet Code Insight User Guide*.

The following sections discuss the downloading, installation, and configuration of the plugins available for various types of build environments:

- [About Scan Agent Plugins](#)
- [Overview of Available Plugins](#)
- [Preparing to Use the Plugins](#)
- [Generating a JWT Authorization Token](#)
- [Downloading Plugins](#)
- [Plugins for Integrated Development Environments \(IDEs\)](#)
- [Plugins for Continuous Integration \(CI\) Tools](#)
- [Plugins for Package Managers and Build Tools](#)
- [Plugins for Binary Repositories](#)
- [Plugins for Container Platforms](#)
- [Generic Scan Agent Plugin](#)

## About Scan Agent Plugins

Once a FlexNet Code Insight scan agent plugin is installed and the scan is configured as part of your build process, the scan agent, when run, collects and sends the scan results back to Code Insight server as published inventory awaiting review, management, and remediation through Code Insight user interface. As with published inventory generated by the Code Insight scan server, published inventory generated by a scan agent plugin can be automatically reviewed by license or security policies as part of the scan and, for inventory not reviewed by policy, can be reviewed manually by legal or security experts. Security alerts with corresponding email notifications will be generated for any inventory item with new security vulnerabilities.

## Overview of Available Plugins

FlexNet Code Insight provides the following plugins that enable data (codebase files) on remote servers to be scanned:

**Table 2-1 • Overview of the Standard Plugins**

Build Environment	Code Insight Plugin	Performs automated scanning of...
<b>IDEs</b>	Eclipse	An Eclipse workspace in the Eclipse IDE environment.
<b>CI Tools</b>	Azure DevOps	An Azure DevOps workspace as part of the build process.
	Bamboo	A Bamboo workspace as part of the build process (on Local Agents only)
	GitLab	GitLab projects as part of the build process.
	Jenkins	A Jenkins workspace as part of the build process.  A separate plugin is available (called the Scan Schedule Plugin) that enables you to simply schedule the scan of a codebase residing on the Code Insight scan server via the Jenkins scheduler.
	TeamCity	TeamCity projects as part of the build process.
<b>Package Manager and Build Tools</b>	Ant	Apache Ant as part of the build process.
	Gradle	Gradle projects as part of the build process.
	Maven	Maven projects as part of the build process.
<b>Binary Repositories</b>	JFrog Artifactory	Artifactory repositories to identify non-compliant artifacts.
<b>Container Platforms</b>	Docker Images	Docker images on a Docker server.

Additionally, a generic scan agent plugin is available with Code Insight that enables you to scan arbitrary file systems of your choice. It also easily integrates with certain Engineering systems, such as TeamCity and GitLab, to perform scans as part of a build process or can serve as an example for developing your own scan agent plugin (as described in the chapter [Developing Custom Plugins](#)).

All the scan agent plugins send results to FlexNet Code Insight for inventory, review, management, remediation, and security alerting.

# Preparing to Use the Plugins

Before configuring and using the scan agent plugins, complete the following tasks:

- Ensure that the FlexNet Code Insight server is installed and running, as described in the *FlexNet Code Insight Installation & Configuration Guide*. (Take note of the server's URL, as you will need this information to configure the plugin to access the server.)
- Generate a JSON Web Token (JWT) for a user registered on the FlexNet Code Insight server. See [Generating a JWT Authorization Token](#) for details.
- Create a project of the type **Inventory Only** on the FlexNet Code Insight server. See "Creating a Project" in the *FlexNet Code Insight User Guide* for details.

## Generating a JWT Authorization Token

FlexNet Code Insight uses a JSON Web Token (JWT) to authorize user access to the public REST API. Several of the scan agent plugins make use of the REST API, so they require a JWT. The following procedure explains how to generate and specify a JWT for Code Insight.



### Task

**To generate a JWT authorization token, do the following:**

1. Log into FlexNet Code Insight.
2. Open the **Admin** menu and select **Preferences**. The **Preferences** page appears.
3. In the **AUTH Tokens** pane, select to create a new JWT or copy an existing one:
  - To copy an existing JWT, select the token from the list, and click the **clipboard** icon (📋) found in the **Actions** column. A dialog is displayed, requesting that you click the **Select Token Text** button and press CTRL-C to copy token text to the clipboard.
  - To create a new JWT, click **Add Token**. The **Add Token** dialog is displayed, requesting that you enter a name for the new token and specify an expiration date (or choose **Never Expires**). Then click **Save**.
4. Copy the new token string, and paste it into the **Token** field on the **FlexNet Code Insight Scan** dialog to configure the desired scan agent plugin (Jenkins, Bamboo, and so forth).

## Downloading Plugins

The scan agent plugins are provided in a .zip file that is not included with the Code Insight installation. You can access the plugins .zip file from Flexera's Customer Community page. The following procedure assumes you have a login and password to access the Customer Community page on [Flexera.com](https://flexeracommunity.force.com/customer/).



### Task

**To download the plugin zip file, do the following:**

1. Log into the Customer Community page on the Flexera website:  
<https://flexeracommunity.force.com/customer/>

2. Click **Downloads**.
3. Click the **Access** button under **FlexNet Code Insight**. The Product and License Center page appears.
4. Select **FlexNet Code Insight** from the **Your Downloads** list.
5. Select the version of FlexNet Code Insight from the list. The **Downloads** page appears.
6. Download the `CodeInsightversionPlugins.zip` file.
7. When the download finishes, extract and copy the desired plugin subfolder to your build environment (or the location that this document might specify for a particular plugin):
  - For a standard scan agent plugin (such as Ant, Artifactory, Bamboo, Docker, Gradle, Jenkins, Maven, and others), extract the subfolder that identifies the plugin (such as `code-insight-docker-images-plugin` for the Docker Images plugin).
  - For the Code Insight generic scan agent plugin (required for Team City or GitLab scans), extract the subfolder `code-insight-agent-sdk-generic-plugin`. This plugin can also be used to scan arbitrary files or can serve as a basis for developing custom scan agent plugins.

Ensure that you copy the entire subfolder to your build location, so you have all necessary files to implement the plugin.

## Plugins for Integrated Development Environments (IDEs)

Currently, FlexNet Code Insight support for scan integration with an integrated development environments (IDE) includes the [Eclipse Plugin](#).

### Eclipse Plugin

The Eclipse plugin enables development teams to perform Code Insight scans within their Eclipse IDE environment. The scan results are automatically sent to the Code Insight server for inventory review, management, remediation, and security-alerting through the Code Insight user interface.

To enable this functionality, you need to install the Eclipse plugin and configure it for your Eclipse project:

- [Prerequisites for the Eclipse Plugin](#)
- [Installing the Eclipse Plugin](#)
- [Configuring the Eclipse Plugin](#)
- [Running a Scan within Your Environment](#)
- [Uninstalling the Eclipse Plugin](#)

### Prerequisites for the Eclipse Plugin

Before you can install and configure the FlexNet Code Insight plugin for Eclipse, perform the required tasks described in [Preparing to Use the Plugins](#) for details.

## Installing the Eclipse Plugin

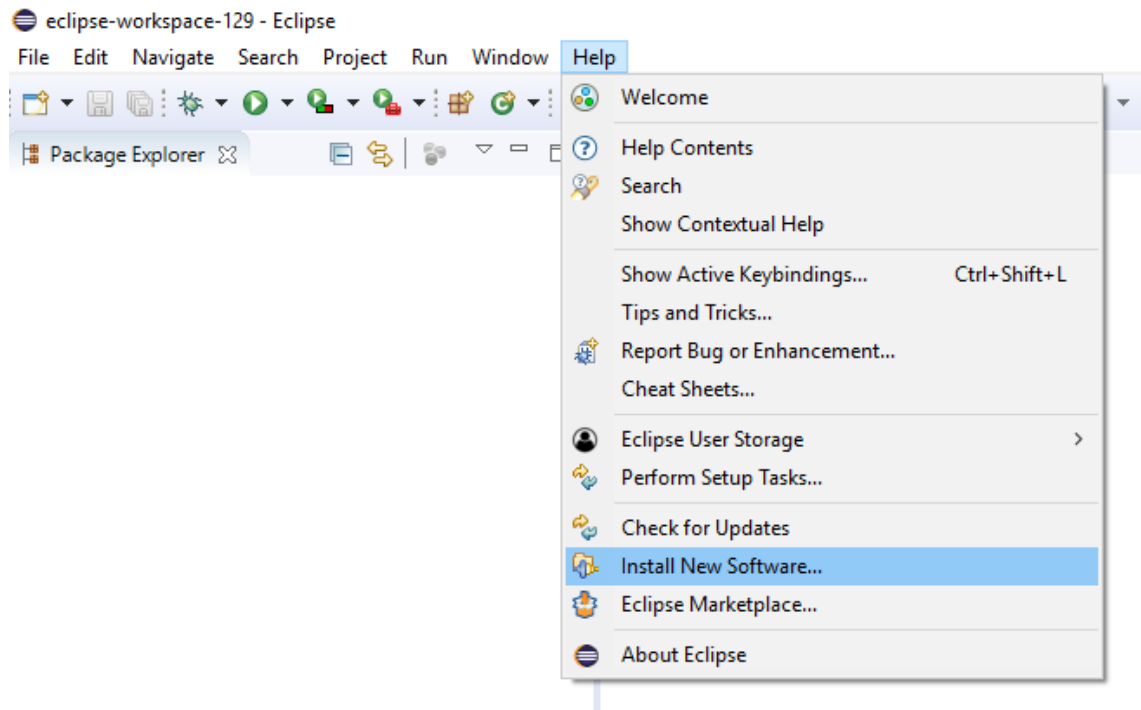
Once you have ensured that the prerequisites are met, use this procedure to install the Eclipse plugin in your Eclipse environment.



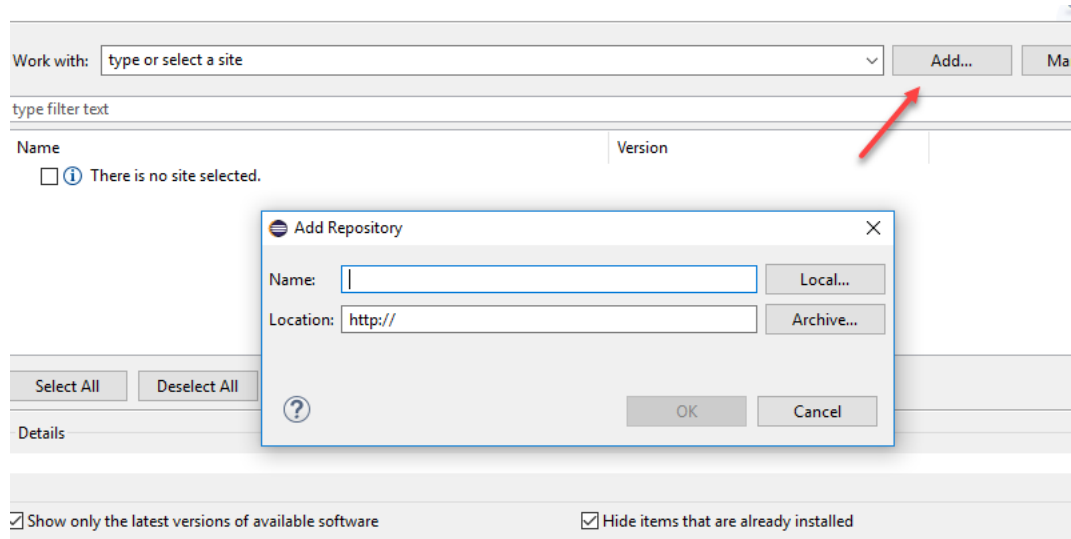
### Task

**To install the Eclipse plugin, do the following:**

1. Ensure that you have extracted the `code-insight-eclipse-scan` folder from the `CodeInsightversionPlugins.zip` file and have placed it in a location accessible to your environment. For more information, see [Downloading Plugins](#).
2. Open Eclipse, and select **Help | Install New Software**.

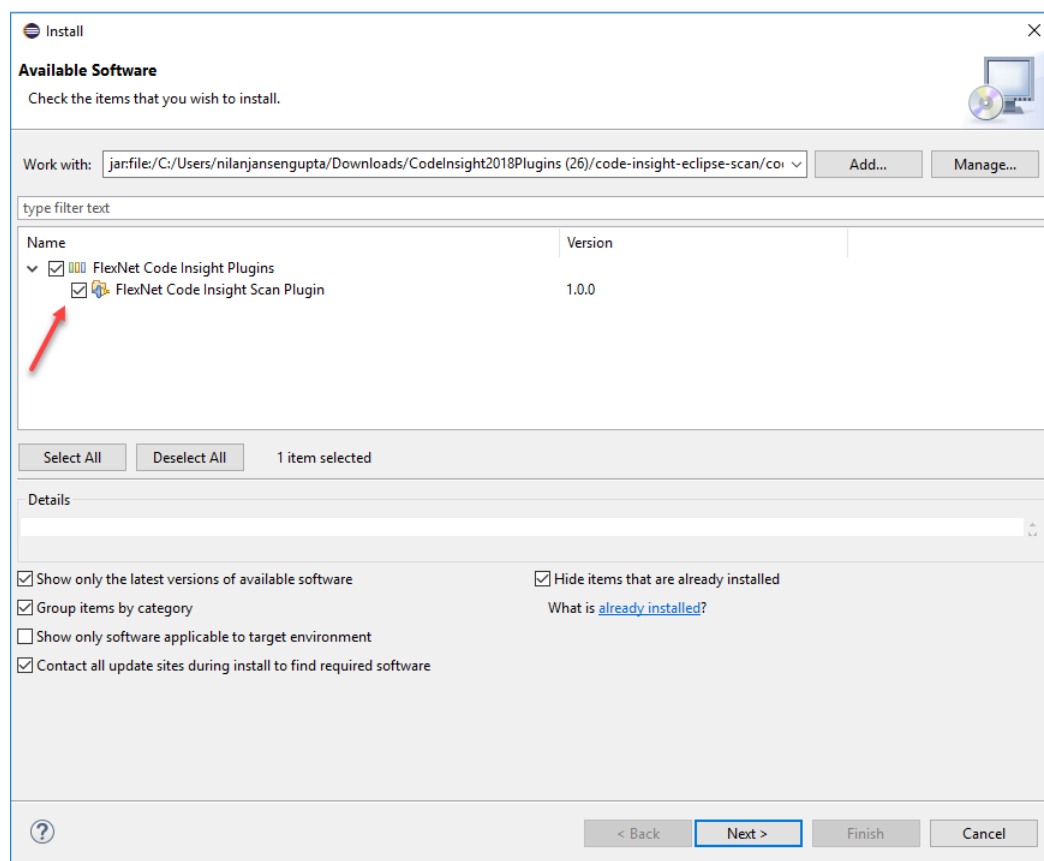


- On the **Available Software** window, click the **Add** to display the **Add Repository** popup:

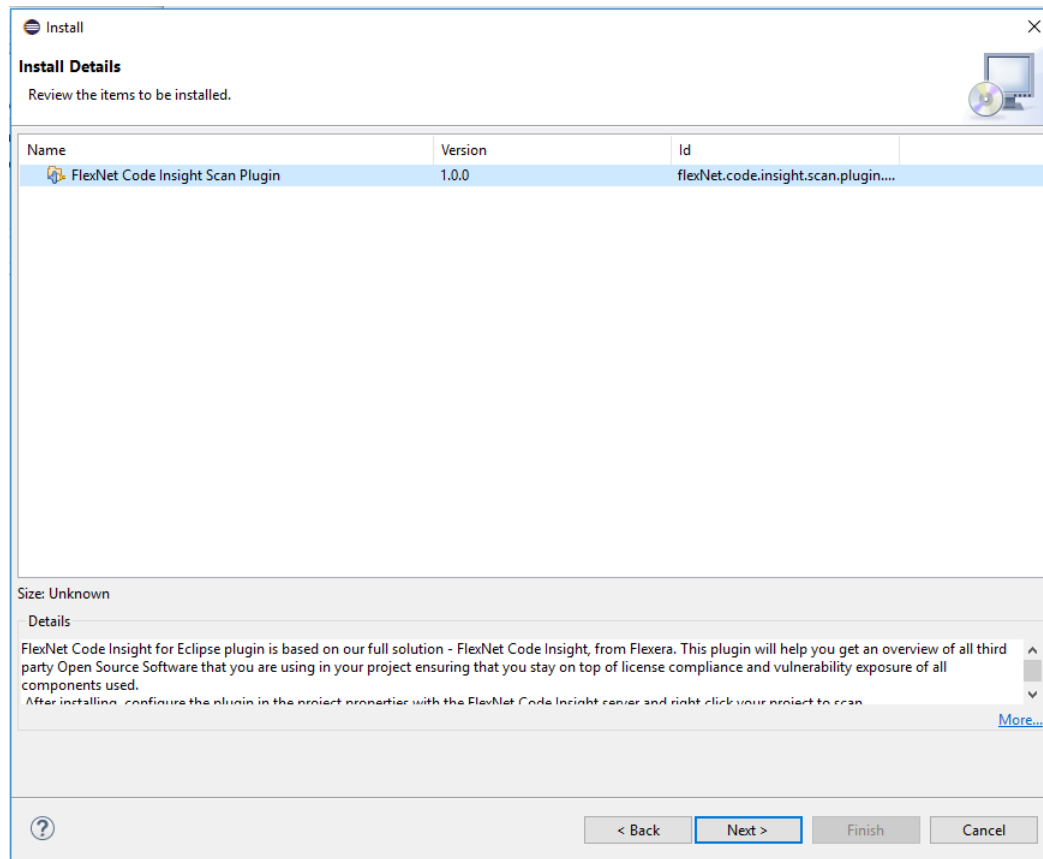


- On the popup, click **Archive**.
- Browse to the `code-insight-eclipse-scan/code-insight-eclipse-scan.zip` file, select it, and click **Open**. Then click **OK**.

The **Available Software** window is redisplayed, showing the plugin information you selected:



- Click the checkbox next to the **FlexNet Code Insight Scan Plugin** entry, and click **Next**. The **Install Details** window is displayed, listing the plugin you are preparing to install.



- Click **Next** to display the **Review Licenses** window.
- Accept the license agreement terms, and click **Finish**.
- When the installation is completed, restart Eclipse.

## Configuring the Eclipse Plugin

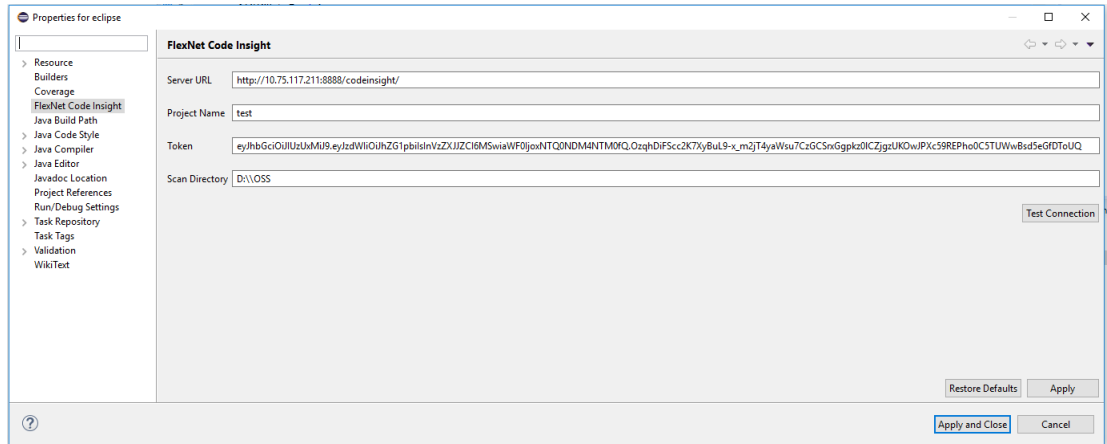
Once you have installed the Eclipse plugin and have restarted Eclipse, follow this procedure to configure the plugin to perform codebase scans on your project in Eclipse.



### Task

**To configure the Eclipse plugin, do the following:**

- In Eclipse, select the project whose codebase you want to scan, right-click the project entry, and select **Properties**.
- On the **Properties** window for your project, select **FlexNet Code Insight** in the left pane to display the properties needed for a Code Insight scan.



- 3.** In the **Properties** window for your project, provide the following property values:

Field	Description
Server URL	The URL for the Code Insight core server (for example, <a href="http://fnciserver.myorg.org:8888/codeinsight/">http://fnciserver.myorg.org:8888/codeinsight/</a> ). Ensure that the URL is publicly accessible and that the port is available.
Project Name	The name of the inventory-only project that was created in the Code Insight user interface (for example, <a href="#">ScanProject2_eclipse</a> ).
Token	The JWT authorization token generated using the Code Insight user interface. (Copy and paste the token into this field.) See <a href="#">Generating a JWT Authorization Token</a> .
Scan Directory	The folder containing the code to scan (for example, <a href="#">C:\Users\user1\eclipse-workspace\project2_code</a> ).

- Click **Test Connection** to ensure you can connect to the FlexNet Code Insight server. A message box is displayed, indicating whether the test is successful. If the test is unsuccessful, adjust the property values as needed and retest the connection.
- When the plugin has been properly configured, click **Apply and Close**.

## Running a Scan within Your Environment

Once the Eclipse plugin has been properly configured, you can invoke a scan on your codebase.

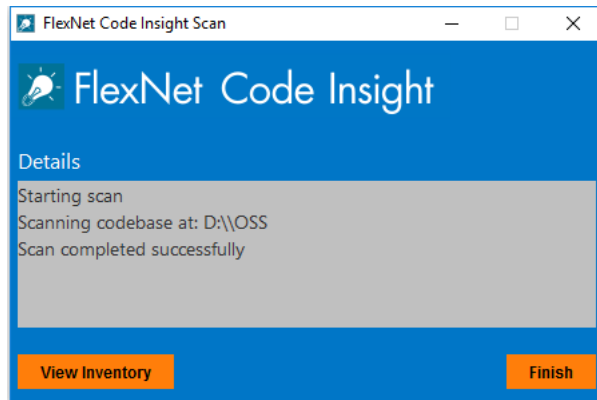




**Task**

**To run a scan on your codebase, do the following:**

1. In Eclipse, select the project whose codebase you want to scan, right-click the project entry, and select **FlexNet Code Insight | Scan Project**. The FlexNet Code Insight scan window is opened, enabling you to keep track of the scan progress:



2. When the scan completes, do one of the following:
  - Click **Finish** to close the FlexNet Code Insight scan window.
  - Click **View Inventory** to connect to Code Insight, which then opens on the **Project Inventory** tab for the inventory-only project you created for the scan. From here you can begin the review and remediation process of the inventory results. For further instructions, refer to “Using the Project Inventory Tab” in the “Using FlexNet Code Insight” chapter in the *FlexNet Code Insight User Guide*.

## Uninstalling the Eclipse Plugin

Use the following procedure to uninstall the Eclipse plugin.



**Task**

**To uninstall the Eclipse plugin, do the following:**

1. Open Eclipse, and select **Help | About Eclipse**.
2. In the **About Eclipse** window, click **Installation Details**.
3. In the **Eclipse Installation Details** window, select **FlexNet Code Insight Scan Plugin**, and click **Uninstall....**  
The **Uninstall Details** window is displayed, listing the plugin.
4. Click **Finish** to confirm that you want to uninstall the plugin and to start the process.
5. Restart Eclipse when prompted to do so.
6. Navigate to the plugins folder in Eclipse (for example, *eclipse-home/plugins* or *userhome/.p2/pool/plugins*); and, from this folder, remove the *code-insight-eclipse-scan* folder.
7. Navigate to and open the *artifacts.xml* file in Eclipse (for example, *eclipse-home/artifacts.xml* or *userhome/.p2/pool/artifacts.xml*), and remove the *codeinsight-Eclipse-scan* section from the file.

# Plugins for Continuous Integration (CI) Tools

Code Insight provides the following scan agent plugins that integrate with continuous integration (CI) tools:

- [Azure DevOps Extension](#)
- [Bamboo Plugin](#)
- [GitLab Plugin](#)
- [Jenkins Plugin](#)
- [Scan Scheduler Plugin for Jenkins](#)
- [TeamCity Plugin](#)

## Azure DevOps Extension

A FlexNet Code Insight extension for Azure DevOps is available for downloading from the Visual Studio Marketplace. This extension allows development teams to easily integrate Code Insight scanning into their Azure DevOps build process and send scan results to the Code Insight server for inventory review, management, remediation, and security-alerting through the Code Insight user interface.

To enable this functionality, you need to install the extension and configure the build process to include the scan:

- [Prerequisites for the Azure DevOps Extension](#)
- [Installing the Azure DevOps Extension](#)
- [Adding a Scan Task to Your Azure DevOps Agent Job](#)

## Prerequisites for the Azure DevOps Extension

Before you can install and configure the Azure DevOps extension, perform the required tasks described in [Preparing to Use the Plugins](#) for details.

## Installing the Azure DevOps Extension

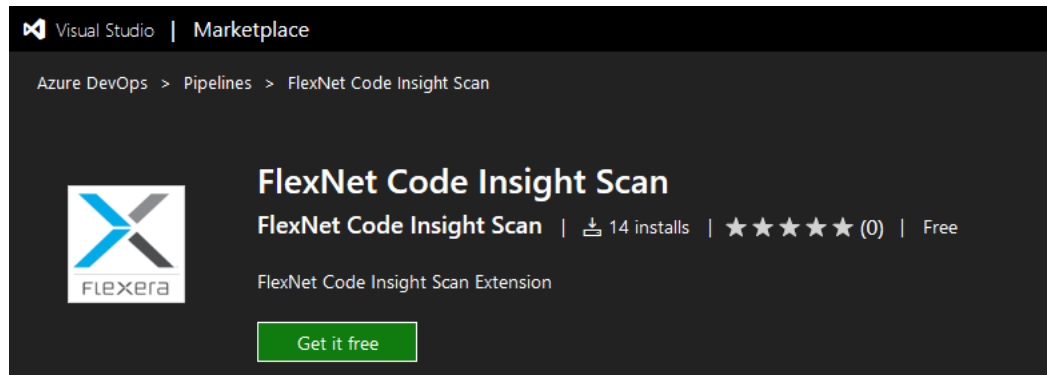
To obtain and install the Azure DevOps extension, perform the following steps.



### Task

**To obtain and install the extension, do the following:**

1. Open the Visual Studio Marketplace:  
<https://marketplace.visualstudio.com/>
2. In the **Azure DevOps** section, search for the **FlexNet Code Insight Scan** extension.



3. Download and install this extension into Azure DevOps.

## Adding a Scan Task to Your Azure DevOps Agent Job

After the Azure DevOps extension has been installed, you need to add a Code Insight scan task to your Azure DevOps agent job so that the scan is automatically performed as part of your build process.



### Task

**To add a scan task to your DevOps agent job, do the following:**

1. Create a build pipeline for your Azure DevOps project.
2. Locate the **FlexNet Code Insight Scan** task in the task catalog.

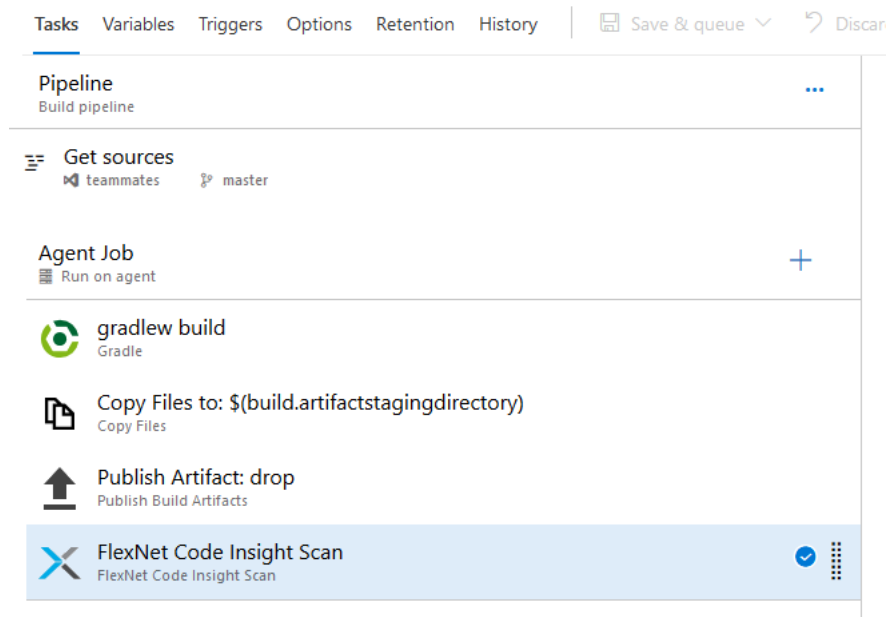
Add tasks | Refresh



#### FlexNet Code Insight Scan

Add this task to your Project to find its inventories and see Reports in FlexNet Code Insight.

3. Add the **FlexNet Code Insight** task at any point after the build task has completed.



4. Define the scan task properties on the **FlexNet Code Insight Scan** window:

FlexNet Code Insight Scan ⓘ [Link settings](#) [View YAML](#) [Remove](#)

Version

Display name \*

FlexNet Code Insight Server \* ⓘ

Authentication Token \* ⓘ

FlexNet Code Insight Project Name \* ⓘ

Folder(s) to Scan ⓘ

Control Options

Output Variables

The following describes the task properties:

Field	Description
<b>FlexNet Code Insight Server</b>	The URL for the core server (for example, <a href="http://fnciserver.myorg.org:8888/codeinsight/">http://fnciserver.myorg.org:8888/codeinsight/</a> ). Ensure that the URL is publicly accessible and that the port is available.
<b>Authorization Token</b>	The JWT authorization token generated using the Code Insight user interface. (Copy and paste the token into this field.) See <a href="#">Generating a JWT Authorization Token</a> .

Field	Description
<b>FlexNet Code Insight Project Name</b>	The name of the inventory-only project that was created in the Code Insight user interface (for example, <code>ScanProject2_AzureDevOps</code> ).
<b>Folder(s) to Scan</b>	The folder containing the code to scan. Typically, you would use <code>\$(build.artifactstagingdirectory)</code> , which is the location where the build output is staged during the build process.

5. Save and queue the build definition.

The scan will be performed in the build environment as part of the build process, and the results will be sent to the project you configured on the Code Insight server. The resulting inventory items can be viewed and managed in the Code Insight user interface.

## Bamboo Plugin

FlexNet Code Insight provides a Bamboo plugin that allows automated scanning of a Bamboo workspace as part of your application build process. The scan results are sent to Code Insight for inventory creation, review, and security alerting.

The Bamboo plugin scans only the application root folder.

The following topics describe how to install and configure this plugin on the Bamboo build server:

- [Prerequisites for the Bamboo Plugin](#)
- [Installing and Configuring the Bamboo Plugin](#)

## Prerequisites for the Bamboo Plugin

Before you install and configure the Bamboo plugin, ensure that the following prerequisites are met:

- Bamboo 5.2 or higher is installed and configured as explained in the Bamboo installation documentation.
- Minimum heap memory size is set to 4GB for the Bamboo Server.
- The Bamboo server instance on which you are running the plugin is a Local Agent. (Scans on Remote Agents are not supported.)
- All FlexNet Code Insight prerequisite tasks for plugins have been performed, as described in [Preparing to Use the Plugins](#).

## Installing and Configuring the Bamboo Plugin

The following procedure covers installing and configuring the Bamboo plugin, which requires you to perform actions in both Bamboo and FlexNet Code Insight.

**Task****To install and configure the Bamboo plugin, do the following:**

1. Extract the Bamboo plugin from the `CodeInsightversionPlugins.zip` file. For more information, see [Downloading Plugins](#).
2. Access your Bamboo server instance.
3. From the **Bamboo Administration icon**, click **Add-ons**.
4. Click **Upload add-on**.
5. Browse to the `code-insight-bamboo-scan.jar` and click **Upload**. The Bamboo jar file is located wherever the zip file containing the plugins was extracted.
6. Create a project in Bamboo:
  - Create the plan.
  - Add a job.
  - Add a FlexNet CodeInsight Scan task to a build project.
7. If you have not done so, create an inventory-only project with a name corresponding to the Bamboo project. For information about creating an inventory-only project, see “Performing Inventory-only Scanning” in the *FlexNet Code Insight User Guide* or online help.
8. Enter the following information in the FlexNet CodeInsight Scan task:
  - **Server URL**—The URL for the core server. For example, `http://fnciserver.myorg.org:8888/codeinsight/`.
  - **Token**—This token must be generated in the FlexNet Code Insight Web UI and pasted into this field. See [Generating a JWT Authorization Token](#).
  - **Project Name**—The name of the project that was created in the FlexNet Code Insight Web UI.
9. Click **Save**. If the Server URL and Token value are correct, the task will be saved. The next time you run the plan, the automated scan of the workspace will be executed for the configured project as part of the plan.



**Note** • The scan task should be placed after the build task in the plan’s task sequence.

## GitLab Plugin

This section explains how to configure GitLab to integrate with the FlexNet Code Insight generic scan agent plugin to perform an automatic composition scan as part of the build process in a Windows environment. The following topics are covered. Note that the scan requires the generic scan agent plugin.

- [Prerequisites for the GitLab Plugin](#)
- [Installing the Generic Scan Agent on GitLab Runner](#)
- [Configuring the CI/CD Pipeline to Run a Scan](#)
- [Executing the Build](#)

## Prerequisites for the GitLab Plugin

The following prerequisites are required to integrate GitLab with the FlexNet Code Insight generic scan agent plugin:

- A GitLab runner properly installed on Windows. (Refer to <https://docs.gitlab.com/runner/install/> for instructions.)
- All the prerequisites listed in [Prerequisites for the Generic Scan Agent Plugin](#).
- The generic scan agent plugin (as described in [Installing the Generic Scan Agent on GitLab Runner](#)).

## Installing the Generic Scan Agent on GitLab Runner

The FlexNet Code Insight generic scan agent plugin is located in the `code-insight-agent-sdk-generic-plugin` folder, extracted from the `CodeInsightVersionPlugins.zip` file (see [Downloading Plugins](#)). Copy this folder, containing a jar file and a sample batch and shell script, to the GitLab runner installed on Windows.

On the GitLab runner, update the following to match your environment:

- The codebase root:  

```
SET ROOT_PATH=C:\GitLab-Runner\output
```
- The bin folder location for the generic scan agent plugin:  

```
cd C:\GitLab-Runner\GenericScanPlugin\example\bin
```

Note that the first time a scan is performed using the generic scan agent plugin, a data snapshot is downloaded from the National Vulnerability Database (NVD) to generate an index of the latest security vulnerabilities.

## Configuring the CI/CD Pipeline to Run a Scan

To configure the CI/CD pipeline in your GitLab project to run a Code Insight scan, you need to edit your `.gitlab-ci.yml` file.



### Task

**To edit the `.gitlab-ci.yml` file, do the following:**

Add the following contents to the file:

```
variables:
  FNCI_SERVER: <FNCI_SERVER>
  FNCI_TOKEN: <AUTH_TOKEN>
  FNCI_PROJECT: <FNCI_PROJECT_NAME>

codeinsight_scan:
  stage: test
  only:
    - master
  tags:
    - <tag for your GitLab-Runner>
  script:
    - cmd /Q /C C:\Gitlab-runner\GenericScanPlugin\example\bin\run_scan.bat %FNCI_PROJECT%
      %FNCI_SERVER% %FNCI_TOKEN% %CI_PROJECT_DIR%
```

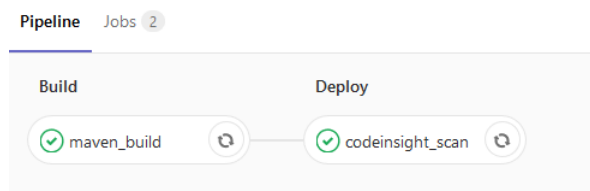
Replace the following variables with the appropriate information:

- <FNCCI\_SERVER> with your Code Insight server (for example, <http://1.1.1.1:8888/codeinsight>)
- <AUTH\_TOKEN> with your JWT authorization token obtained from the Code Insight server (as described in [Generating a JWT Authorization Token](#))
- <FNCCI\_PROJECT\_NAME> with the project you created earlier to capture the inventory
- <tag for your GitLab-Runner> with the tag of your GitLab runner

%CI\_PROJECT\_DIR% is the GitLab variable for the project path where the code is built. You can replace it with the path of the folder containing the binaries of your built project.

## Executing the Build

The next time your build is executed, a FlexNet Code Insight agent scan is performed at the end of the build process. If you have scheduled the Code Insight scan job after a Maven build, for example, you should see something like this in your GitLab pipeline:



## Jenkins Plugin

FlexNet Code Insight provides a Jenkins plugin that enables automated scanning of the Jenkins workspace as part of the build process or Jenkins Pipeline process. The scan results are sent to FlexNet Code Insight for inventory creation, review, and security alerting.

The Jenkins plugin installation and configuration process proceeds in the following manner:

**Phase 1**—Address the prerequisites for the Jenkins plugin. See [Prerequisites for the Jenkins Plugin](#).

**Phase 2**—Set the heap size. See [Setting Heap Size for the Jenkins Plugin](#).

**Phase 3**—Set up the Jenkins plugin. See [Setting Up the Code Insight Jenkins Plugin](#).

For examples on how to include the Code Insight scan as a part of a Jenkins Pipeline, see [Support for the Jenkins Pipeline](#).

## Prerequisites for the Jenkins Plugin

Before you install and configure the Jenkins plugin, ensure that the following prerequisites are met:

- Ensure that Jenkins is installed and configured properly in your environment.
- Perform the required FlexNet Code Insight tasks for plugins, as described in [Preparing to Use the Plugins](#).



## Setting Heap Size for the Jenkins Plugin

The Jenkins plugin requires a minimum of 4GB heap for scanning. The heap size may need to be adjusted based on the number of parallel scans to be executed. In addition, ensure that you are using a 64-bit Java virtual machine (JVM) and that you run the scan agent as a Jenkins Slave, which is a Java executable that usually runs on a remote machine. The procedure for setting the heap size differs depending upon the environment you are using, Windows or Linux. Follow the procedure for your environment.

### On Windows



**Task** *To set the heap size in Windows, do the following:*

1. Open the `jenkins.xml` configuration file.
2. Update the `<EXECUTABLE>` value to point to your 64-bit JVM:  
`<EXECUTABLE>C:\Java\jdk1.8\jre\bin\java</EXECUTABLE>`
3. Update the JVM arguments (`-Xmx` value) to allocate a minimum heap size of 4GB:  
`<ARGUMENTS>-Xrs -Xmx4g -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -jar  
 "%BASE%\jenkins.war" --httpPort=8080 --webroot="%BASE%\war"</ARGUMENTS>`



**Note** • The heap size may have to be adjusted based on the number of parallel scans to be executed.

### On Linux



**Task** *To set the heap size in Linux, do the following:*

1. Open the `/etc/default/jenkins` file.
2. Update the JVM arguments to allocate a minimum heap size of 4GB:  
`JAVA_ARGS="-Xmx4096m"`



**Note** • The heap size may have to be adjusted based on the number of parallel scans to be executed.

## Setting Up the Code Insight Jenkins Plugin

The following procedure describes how to configure the Jenkins plugin.

**Task**

**To set up the Jenkins plugin, do the following:**

1. Extract the Jenkins plugin from the `CodeInsightversionPlugins.zip` file. For more information, see [Downloading Plugins](#).
2. Access your Jenkins server instance and navigate to **Manage Jenkins -> Manage Plugins -> Advanced tab -> Upload Plugin**.
3. Browse to the `code-insight-scan-plugin.hpi` file, and click **Upload**.
4. Restart the Jenkins server after installing the plugin.
5. Create a new Jenkins project:
  - a. Click **New Item**.
  - b. Enter a name.
  - c. Select a project type.
  - d. Click **OK**.
6. To configure the project, select **Add post-build action** from the Post-build action dropdown menu, and select **Scan with FlexNet Code Insight**. The **Scan with FlexNet Code Insight** dialog appears.
7. Enter the following information in the **Scan with FlexNet Code Insight** dialog:
  - **Code Insight Core Server Base URL**—The URL for the core server (for example, `http://fnciserver.myorg.org:8888/codeinsight`).
  - **User Access Token**—This token generated using the FlexNet Code Insight user interface. Copy and paste the token into this field. See [Generating a JWT Authorization Token](#).
  - **Project Name**—The name of the project that was created in the FlexNet Code Insight user interface (for example, `ScanMaster_WindowsJenkins1`).
8. Click **Test Connection** to test your connection to FlexNet Code Insight.
9. Click **Save**. The next time you build, the scan will be performed after the build action.



**Note** • Ensure that your Jenkins server environment has a minimum of 4GB of heap space, adjusted based on the number of parallel scan to be executed. Also ensure that the environment is configured with a 64-bit JRE to support that amount of heap space. In addition, run the scan agent as a Jenkins Slave.

## Support for the Jenkins Pipeline

The Code Insight plugin for Jenkins supports the inclusion of the Code Insight scan in a Jenkins Pipeline, as described in the following topics:

- [Providing the Pipeline Script for the Scan Step](#)
- [Pipeline Code Examples for Running the Scan](#)

## Providing the Pipeline Script for the Scan Step

Once you build the Pipeline job, you need to include the Pipeline script for the scan step, StartScan, in your Pipeline code. (The next section, [Pipeline Code Examples for Running the Scan](#), provides examples of Pipeline code that include this script.)

To create the Pipeline script for the StartScan step, you can use one of these methods:

- Go to the Snippet Generator, select the **StartScan: Scan workspace and send results to FlexNet Code Insight** step, and generate the script. Then copy and paste the generated script into the Pipeline code.
- Simply create the script for the StartScan step as highlighted in the Pipeline code examples.

See [Setting Up the Code Insight Jenkins Plugin](#) for a description of the properties (base URL, project name, and JWT token) used in the Pipeline script.

## Pipeline Code Examples for Running the Scan

Jenkins supports two syntax types for the development of Pipeline code:

- **Scripted syntax**—The “traditional” syntax used to develop the Pipeline as a script using Groovy as the domain-specific language.
- **Declarative syntax**—A simple, user-friendly syntax with a predefined hierarchy of statements that makes Pipeline development easier than with the Scripted syntax. Additionally, it does not require knowledge of the Groovy language. Jenkins support for the Declarative syntax was introduced with Jenkins Pipeline Plugin 2.5.

The following examples show both types of Pipeline code syntax in which the Pipeline script for the scan has been incorporated:

- [Example Declarative Pipeline Code to Run the Scan](#)
- [Example Scripted Pipeline Code to Run the Scan](#)

The Pipeline script for the scan step is highlighted in each example.

### Example Declarative Pipeline Code to Run the Scan

The following is an example of Declarative code used to run the Code Insight scan as a StartScan step in the Pipeline process:

```
pipeline {
  agent any
  stages {
    stage('Checkout build and scan project1') {
      steps {
        git credentialsId: 'abcd', url: 'git://git.company.com/organization/repository1.git'
        sh "'PATH_TO_MAVEN/bin/mvn' clean install"

        StartScan (baseUrl: 'http://HOST_NAME:PORT/', projectName: 'FNCI_PROJECT_NAME',
                    token: 'JWT_TOKEN')
      }
    }
  }
}
```

## Example Scripted Pipeline Code to Run the Scan

The following is an example of Scripted Pipeline code used to run the Code Insight scan as a `StartScan` step in the Pipeline process. The example also shows how to set up individual scans within a single Pipeline job by specifying multiple directories.

```
node {
  checkout1()
  checkout2()
}

def checkout1(){
  dir("project-1"){
    stage ('Checkout project 1'){
      git credentialsId: 'abcd', url: 'git://git.company.com/organization/repository1.git'
    }
    stage ('Build Project 1'){
      build()
    }
    stage ('Scan Project 1'){
      StartScan (baseUrl: 'http://HOST_NAME:PORT/', projectName:
        'FNCI_PROJECT_NAME', token: 'JWT_TOKEN')
    }
  }
}

def checkout2(){
  dir("project-2"){
    stage ('Checkout project 2'){
      git credentialsId: 'abcd', url: 'git://git.company.com/organization/repository2.git'
    }
    stage ('Build Project 2'){
      build()
    }
    stage ('Scan Project 2'){
      StartScan (baseUrl: 'http://HOST_NAME:PORT/', projectName: 'FNCI_PROJECT_NAME',
        token: 'JWT_TOKEN')
    }
  }
}

def build(){
  sh "'PATH_TO_MAVEN/bin/mvn' clean install"
}
```

## Scan Scheduler Plugin for Jenkins

The Jenkins Scan Scheduler plugin enables you to simply schedule the scan of a codebase residing on the FlexNet Code Insight scan server via the Jenkins scheduler. Before you install and configure the Jenkins Scan Scheduler plugin, ensure that the following prerequisites are met:

- Jenkins must be installed and configured properly in your environment.
- The project of interest must be set up in FlexNet Code Insight. For information on creating a FlexNet Code Insight project, see “Creating a Project” in the *FlexNet Code Insight User Guide*.



**Task**

**To install the FlexNet Code Insight Scan Scheduler for Jenkins, do the following:**

1. Sign into Jenkins CI.
2. Navigate to **Manage Jenkins > Manage Plugins > Advanced**. The **Upload Plugin** dialog appears.
3. Click **Choose File** and select the `code-insight-scan-scheduler.hpi` file.
4. Click **Upload**.
5. Restart the Jenkins server after uploading the plugin.
6. Create a new Jenkins project:
  - Click **New Item**.
  - Enter a name.
  - Select a project type.
  - Click **OK**.
7. To configure the project, select **Add build step** from the **Build** dropdown menu, and select **Schedule a Code Insight Scan**. The **Schedule a Code Insight Scan** dialog appears.
8. Enter the following information in the **Schedule a Code Insight Scan** dialog:
  - **Server URL**: The URL for the core server. For example, `http://fnciserver.myorg.org:8888/codeinsight/`.
  - **Token**: This token must be generated in the FlexNet Code Insight Web UI and pasted into this field. See [Azure DevOps Extension](#).
  - **Project ID**: The ID of the project that was created in the FlexNet Code Insight Web UI.
9. Click **Test Connection** to test your connection to FlexNet Code Insight.
10. Click **Save**. The next time you build, the scan will be scheduled on the FlexNet Code Insight server for the configured project as part of the build.

## TeamCity Plugin

This section explains how to configure TeamCity to integrate with the Code Insight's generic scan agent plugin to perform an automatic composition scan as part of the build process. The following topics are covered. Note that the scan requires the generic scan agent plugin.

- [Prerequisites for TeamCity Plugin](#)
- [Installing the Generic Scan Agent on the TeamCity Agent](#)
- [Configuring a Build to Run a Code Insight Scan](#)
- [Executing the Build](#)

## Prerequisites for TeamCity Plugin

The following prerequisites are required to integrate TeamCity with the Code Insight generic scan agent plugin:

- A TeamCity build agent properly installed. (Refer to <https://confluence.jetbrains.com/display/TCD10/Setting+up+and+Running+Additional+Build+Agents> for instructions.)
- All the prerequisites listed in [Prerequisites for the Generic Scan Agent Plugin](#).
- The generic scan agent plugin (as described in [Installing the Generic Scan Agent on the TeamCity Agent](#)).

## Installing the Generic Scan Agent on the TeamCity Agent

The Code Insight generic scan agent plugin is located in the `code-insight-agent-sdk-generic-plugin` folder, extracted from the `CodeInsightVersionPlugins.zip` file (see [Downloading Plugins](#)). Copy this folder, containing a jar file and a sample batch and shell script, to the TeamCity build agent.

On the Team City build agent, update the following to match your environment:

- The codebase root:  

```
SET ROOT_PATH=C:\Codebase\output
```
- The bin folder location for the generic scan agent plugin:  

```
cd C:\agent\GenericScanPlugin\example\bin
```

Note that the first time a scan is performed using the generic scan agent plugin, a data snapshot is downloaded from the National Vulnerability Database (NVD) to generate an index of the latest security vulnerabilities.

## Configuring a Build to Run a Code Insight Scan

Follow these steps to configure a build that runs a Code Insight scan.



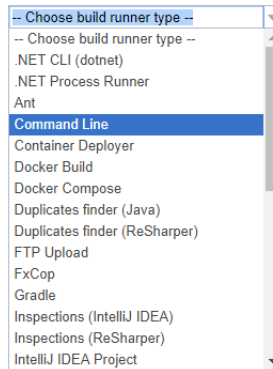
### Task

**To configure a build to run a Code Insight scan, do the following:**

1. Log into TeamCity, select your project, and create a new Build Configuration.
2. To configure a build step to run a Code Insight scan, select on your build configuration, and click **Add Build Step**.
3. From the **Runner type** list, select **Command Line**.

New Build Step: | ▾

Runner type:



4. Configure the **Command line** build step for the Code Insight scan:

- a. Enter a value for **Step name** (for example, **Codeinsight Scan**) to identify the step.
- b. In the **Run** field, select **Custom script**.
- c. In the **Custom script** field, provide the following:

```
C:\GenericScanPlugin\example\bin\TeamCity_FNCIScan.bat <FNCI_PROJECT_NAME> <FNCI_SERVER>
<JWT_TOKEN> <SCAN_DIR>
```

Replace the following variables in the script with the appropriate information:

- <FNCI\_PROJECT> with the name of the Code Insight project you created earlier to capture the inventory
- <FNCI\_SERVER> with your Code Insight server URL (for example, <http://1.1.1.1:8888/codeinsight>)
- <JWT\_TOKEN> with your JWT obtained from the Code Insight server (as described in [Generating a JWT Authorization Token](#))
- <SCAN\_DIR> with the directory that you want to scan

Build Step

[+ Add build step >](#)

Runner type:

Command Line

Simple command execution

Step name:

Code Insight Scan

Optional, specify to distinguish this build step from other steps.

Run:

Custom script

Custom script:

Enter build script content:

C:\GenericScanPlugin\example\bin\TeamCity\_FNCIScan.bat

A platform-specific script, which will be executed as a .cmd file on Windows or as a shell script in Unix-like environments.

[Show advanced options](#)

Save

Cancel

When complete, your build configuration should look like this:

**Build Steps**  
In this section you can configure the sequence of build steps to be executed. Each build step is represented by a build runner and provides integration with a specific build or test tool.

[+ Add build step](#)

Build Step	Parameters Description		
1. Code Insight Scan	Command Line Custom script: C:\GenericScanPlugin\example\bin\TeamCit... Execute: If all previous steps finished successfully	<a href="#">Edit</a>	<a href="#">...</a>

## Executing the Build

The next time your build is executed, a scan will be performed at the end of the build process. If you have scheduled the Code Insight scan job, after a Maven build, for example, you should see something like below in your TeamCity build queue:

Maven Code Insight Scan				no hidden	
Build				Run	
#2	Tests passed: 836	No artifacts	38851806+vdonga (1)	9 minutes ago (1m:52s)	
Code Insight Scan				Run	
#1	Success	No artifacts	No changes	5 minutes ago (4m:36s)	

## Plugins for Package Managers and Build Tools

Code Insight provides the following scan agent plugins that integrate with package manager and build tools:

- [Apache Ant Plugin](#)
- [Gradle Plugin](#)
- [Maven Plugin](#)

## Apache Ant Plugin

Apache Ant is a tool to support the build process for Java projects. Ant is often used in conjunction with other build tools such as Maven. Code Insight provides the Apache Ant plugin to run a scan task along with the target of the build cycle and send the results, as inventory, to the Code Insight server for review, management, and remediation. The Apache Ant plugin scans only the application root folder.

The following topics describe how to install and configure the Gradle plugin:

- [Prerequisites for the Apache Ant Plugin](#)
- [Configuring the Apache Ant Plugin](#)
- [Executing the Scan](#)

## Prerequisites for the Apache Ant Plugin

Before you install and configure the Apache Ant plugin, ensure that the following items are correctly installed and configured:

- JDK 1.8



- Apache Ant

Also ensure that all FlexNet Code Insight prerequisite tasks for plugins have been performed, as described in [Preparing to Use the Plugins](#).

## Configuring the Apache Ant Plugin

Use the following procedure to configure the Apache Ant Plugin to execute along with the build target.



### Task

**To configure the plugin, do the following:**

1. Extract the Ant plugin from the `CodeInsightversionPlugins.zip` file. See [Downloading Plugins](#).
2. Configure `%ANT_HOME%` and add `%ANT_HOME%/bin` to the path variable.
3. To check the Ant plugin installation, run the following command:  
  
`ant -v`
4. Add all the dependent jars from the `code-insight-ant-plugin` folder to the application's compile classpath.
5. To run the task `codeinsightantplugin` along with the compile target, paste the taskdef code snippet into the compile target and run the following command:  
  
`ant compile`  
  
For the code snippet, see [Executing the Scan](#).
6. Copy the `code-insight-ant.jar` into the path used for the compile task, and set the `classpath_refid` of the `javac` task as the `classpathref` in the `codeinsightantplugin` task.

## Executing the Scan

Use the following procedure to execute the scan along with the build target.



### Task

**To execute the scan, do the following:**

1. Run the following command:  
  
`ant targetname`  
  
For example: `ant compile`
2. To execute the scan along with any target of the build lifecycle, apply the plugin inside the target in the `build.xml` of the Ant application as follows:  
  

```
<taskdef name="codeinsightantplugin" classname="com.ant.plugin.CodeInsightAntPlugIn"
classpath=" " classpathref=" " />
<codeinsightantplugin fnciServer="<SERVER_URL>" fnciauthtoken="<BEARER_SERVER_TOKEN_VALUE>"
fnciprojectname="<CODE_INSIGHT_PROJECT_NAME>"
pluginRootPath="<PLUGIN_ROOT_PATH>"
pluginProjectName="<APPLICATION_PROJECT_TO_SCAN>"
pluginindescription="<APPLICATION_DESCRIPTION>"
```

```
pluginPathPrefix="<PLUGIN_PATH_PREFIX>">
</codeinsightantplugin>
```

See descriptions of these settings in [Installing and Configuring the Gradle Plugin](#).



**Note** • The Ant plugin project name can not include the ampersand (&) character.

## Note About “classpath”

Although specifying `taskdef.classpath` is not mandatory, you should set the path id of the `javac` task as the `Classpathref` in the `codeinsightantplugin` taskdef. If the application does not have a `javac` `path-id` defined in the `build.xml`, you must define one new path id referring to all compile time dependencies and use this as `Classpathref`. See the following example:

```
<path id= "cp" <fileset dir="lib">
  <include name="*.jar" />
</fileset>
</path>
```

In this case, use "cp" as the `Classpathref` in the taskdef.

# Gradle Plugin

Gradle is a build automation system that uses the Groovy language to establish the configuration of the build project, rather than using XML as Maven does. The Gradle plugin provided by Code Insight scans a codebase created in Gradle and sends the results, as inventory, to the Code Insight server for analysis, review, and remediation through the user interface. The plugin scans only the following:

- Direct dependencies of a project
- Transitive dependencies of a project
- The distribution folder containing application jars



**Note** • The Gradle plugin does not scan the jars present in the `lib` folder, which contains plugin-dependent jars.

The following topics describe how to install and configure the Gradle plugin:

- [Prerequisites for the Gradle Plugin](#)
- [Installing and Configuring the Gradle Plugin](#)

## Prerequisites for the Gradle Plugin

Before you install and configure the FlexNet Code Insight Gradle plugin, ensure that the following items are correctly installed and configured:

- JDK1.8
- Gradle

Also ensure that all FlexNet Code Insight prerequisite tasks for plugins have been performed, as described in [Preparing to Use the Plugins](#).

## Installing and Configuring the Gradle Plugin

To use the Gradle plugin, you must configure settings in the application's `build.gradle`. This section contains the procedure for installing and configuring the plugin.



### Task

**To install and configure the Gradle plugin, do the following:**

1. Extract the Gradle plugin from the `CodeInsightVersionPlugins.zip` file. See [Downloading Plugins](#).
2. Add all the dependent jars in the `code-insight-gradle-plugin` to the application class path by doing the following:
  - a. Create a folder named `dependent_jars` within the application.
  - b. Copy all jar files into that folder.
  - c. Add the following configuration so that the jars are available to the classpath:

```
buildscript {
    dependencies {
        classpath files(fileTree(dir: 'dependent_jars', includes: ['*.jar']))
    }
}
```

3. Add the following settings to the `build.gradle` file:

```
apply plugin: 'code-insight-gradle-plugin'
```

```
scanSettings {
    fnciServer= "<SERVER_URL>"
    fnciAuthToken= "<BEARER_SERVER_TOKEN_VALUE>"
    fnciProjectName= "<CODE_INSIGHT_PROJECT_NAME>"
    pluginRootPath= "<PLUGIN_ROOT_PATH>"
    pluginProjectName= "<APPLICATION_PROJECT_TO_SCAN>"
    pluginDescription= "<APPLICATION_DESCRIPTION>"
    pluginPathPrefix= "<PLUGIN_PATH_PREFIX>"
}
```

The following is a description of the settings:

- **scanSettings**—An extension to provide the FlexNet Code Insight scan server settings.
- **fnciServer**—(Required) The hosted server where the FlexNet Code Insight application is running.
- **fnciAuthToken**—(Required) The user identification token of the FlexNet Code Insight server containing the command “Bearer”, followed by the token value (as in the example, [Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWUiOiJhZG1pbIIsInVzZXJJZCI6MSwia](#)). For information about generating the Auth token, see [Generating a JWT Authorization Token](#).
- **fnciProjectName**—(Required) The name of the inventory-only Code Insight project existing on the Code Insight server to contains the scan results.
- **pluginRootPath**—(Required) The path where the plugin will be launched, usually the root of the application. An example value is `D:\test\Gradle_test\Gradle_application`. This field is required.

- **pluginProjectName**—(Required) The name of Gradle-based application whose codebase you want to scan.
  - **pluginDescription**—A description of the application to display on the **Summary** tab for the project in Code Insight.
  - **pluginPathPrefix**—The Code Insight server path (for example, [demo\\_workspace/](#)) used as a prefix for codebase file locations, as listed on the **Associated Files** tab for an inventory item in the Code Insight user interface. For example, *demo\_workspace/*. This field is optional.
4. Configure the `code-insight-scan` task to run during or after the build process.

## Maven Plugin

Maven is a tool that simplifies the building and management of Java-based projects. The FlexNet Code Insight Maven plugin allows you to scan an application project during its build on Maven without disrupting the established build process. Once scanned, the codebase can be analyzed in the FlexNet Code Insight user interface. The Maven plugin makes it easy to incorporate scanning and analysis into your development workflow.

For more information, refer to the following:

- [More About the Maven Plugin](#)
- [Prerequisites for the Maven Plugin](#)
- [Installing and Configuring the Maven Plugin](#)
- [Cleaning the Application Project](#)
- [Running the Maven Goal for the Scan](#)

## More About the Maven Plugin

The Maven plugin scans only the following items:

- Direct dependencies of a project
- Transitive dependencies of a project
- Build folder containing the application jars

The plugin creates a Maven goal called `code-insight-scan`, which will be executed along with the *install* phase of the build cycle to get inventory details, as described later in [Running the Maven Goal for the Scan](#).

## Prerequisites for the Maven Plugin

Before you install and configure the Code Insight Maven plugin, ensure that the following prerequisites are met:

- Maven and JDK 1.8 are installed.
- `%MAVEN_HOME%/bin` is configured and added to the path environment variable. (This prerequisite avoids SSL certification issues.)
- All FlexNet Code Insight prerequisite tasks for plugins have been performed, as described in [Preparing to Use the Plugins](#).

## Installing and Configuring the Maven Plugin

Use the following steps to install and configure the Code Insight Maven plugin.



### Task

**To install and configure the Code Insight Maven plugin, do the following:**

1. From the `CodeInsightversionPlugins.zip` file that was downloaded from the Flexera Customer Care Website, extract the Maven plugin subdirectory (`code-insight-maven-plugin`) to a location on your local disk. The recommended location to which to extract this subdirectory is the application project directory.
2. Execute the following commands to install the plugin into the Maven local repository:

```
mvn install:install-file -Dfile="$<PROJECT_DIRECTORY>/code-insight-maven-plugin/lib/code-
insight-maven-scan-<VERSION>.jar" -DpomFile="$<PROJECT_DIRECTORY>/code-insight-maven-
plugin/lib/pom.xml" -DgroupId=com.flexnet.maven -DartifactId=code-insight-maven-scan -
Dversion=<VERSION> -Dpackaging=jar

mvn install:install-file -Dfile="$<PROJECT_DIRECTORY>/code-insight-maven-plugin/lib/
codeinsight-agent-<VERSION>.jar" -DgroupId=com.flexnet.codeinsight -
DartifactId=codeinsight-agent -Dversion=<VERSION> -Dpackaging=jar
```

Note the following variables:

- `<PROJECT_DIRECTORY>` is your application project directory (or the local directory to which you extracted the plugin).
  - `<VERSION>` is the latest version of specific jar file referenced in the command (either the `code-insight-maven-scan` or `codeinsight-agent` jar file).
3. Add the following information to your application `pom.xml` file. Refer to [Plugin and FNCI Server Settings](#) for a description of the values you need to provide for the plugin and `fnciServerSettings` sections.

```
<plugin>
  <groupId>com.flexnet.maven</groupId>
  <artifactId>code-insight-maven-scan</artifactId>
  <version>latest_codeinsight_maven_scan_jar_version</version>
  <inherited>false</inherited>
  <executions>
    <execution>
      <phase>install</phase>
      <goals>
        <goal>code-insight-scan</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <fnciServerSettings>
      <fnciServer>server_url</fnciServer>
      <fnciAuthToken>bearer_server_token_value</fnciAuthToken>
      <fnciProjectName>FNCI_project_name</fnciProjectName>
      <pluginRootPath>plugin_root_path</pluginRootPath>
      <pluginProjectName>plugin_project_name</pluginProjectName>
      <pluginDescription>any_plugin_description</pluginDescription>
      <pluginPathPrefix>plugin_path_prefix</pluginPathPrefix>
    </fnciServerSettings>
  </configuration>
```

</plugin>

## Plugin and FNCI Server Settings

The following describes the settings that you need to define in the `plugin` and `fnciServerSettings` sections of the information you are adding to the application `pom.xml` file (as described in Step 3 of the previous procedure).

**Table 2-2 • FNCI Server Settings in the Application “pom.xml” File**

Setting	Description
<b>version</b>	The version of the <code>code-insight-maven-scan-&lt;VERSION&gt;.jar</code> file included with the current plugin (for example, <b>1.0.2</b> ).
<b>fnciServer</b>	(Required) The URL path to the FlexNet Code Insight server in the following format:  <code>http://&lt;CODE_INSIGHT_SERVER_HOST_NAME&gt;:&lt;PORT_NUMBER&gt;/codeinsight/</code>
<b>fnciAuthToken</b>	(Required) The JSON Web Token (JWT) used to authorize user access to the Code Insight server functionality. The token format includes the command <b>Bearer</b> , followed by the token value, as in the example:  <b>Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsInVzZXJJZCI6MSwia</b>  For information about generating the JWT, see <a href="#">Generating a JWT Authorization Token</a> .
<b>fnciProjectName</b>	(Required) The name of the FlexNet Code Insight inventory-only project created on the Code Insight server for your application codebase scans.
<b>pluginRootPath</b>	Currently not used.
<b>pluginProjectName</b>	(Optional) The name of the <i>application project</i> being scanned. This name will appear, along with the Code Insight project name, in the <b>Last Scan</b> field on the <b>Summary</b> tab for the project in the FlexNet Code Insight user interface. It provides a reference to help a reviewer or developer identify what codebase was scanned.
<b>pluginDescription</b>	(Optional) A description of the application project being scanned. This text will appear in the <b>Description</b> field on the <b>Summary</b> tab for the project in the FlexNet Code Insight user interface.
<b>pluginPathPrefix</b>	(Optional) The path prefix for the codebase files being scanned. This prefix is used to reference the codebase file paths on the <b>Project</b> report generated from the <b>Summary</b> tab for the project in the FlexNet Code Insight user interface.

## Cleaning the Application Project

During a build, Maven can cache an extensive amount of output, which, in turn, can have a negative impact on the performance of the Maven plugin. Therefore, before you run the Maven goal for the Code Insight scan, it is recommended that you clean the application project, a process that clears the cache of the artifacts of previous builds.



---

**Task** *To clean the application project, do the following:*

Execute the following command:

```
mvn clean
```

## Running the Maven Goal for the Scan

After you clean the application project, you can run the `code-insight-scan` Maven goal, which will perform a Code Insight scan on the codebase.



---

**Task** *To execute the goal that runs the Code Insight scan, do the following:*

To build the application (and run the Code Insight scan as part of the build cycle), execute the following command:

```
mvn install
```

Alternatively, to execute the Code Insight scan only, run the specific goal:

```
mvn code-insight-maven-scan:code-insight-scan
```

## Plugins for Binary Repositories

Currently, FlexNet Code Insight support for scan integration with binary repositories includes the [JFrog Artifactory Plugin](#).

### JFrog Artifactory Plugin

JFrog Artifactory is a binary repository manager where third-party artifacts are stored. The Artifactory repository is centralized, so all developers use the same repository to access artifacts, which provides faster access, control, and security of binary artifacts. The Artifactory plugin provided by Code Insight scans an Artifactory repository and sends the results, as inventory, to the Code Insight server for review, management, and remediation through the user interface. Because Artifactory can contain several repositories, the plugin can support a multiple-repository scan, generating inventory for each repository in a separate Code Insight project.

The following topics describe how to install and use the Artifactory plugin:

- [Prerequisites for the Artifactory Plugin](#)
- [Installing the Artifactory Plugin](#)
- [Scanning an Artifactory Repository Using a Cron Job](#)
- [Scanning an Artifactory Repository Using REST API](#)

- Scan Results

## Prerequisites for the Artifactory Plugin

Before installing and using the Artifactory plugin, ensure that the following prerequisites are met:

- Your site uses JFrog Artifactory PRO 5.x or higher.
- All FlexNet Code Insight prerequisite tasks for plugins have been performed, as described in [Preparing to Use the Plugins](#).
- You have write access for the `etc/plugins` directory on the Artifactory server. If you do not have access to that directory, be sure to obtain access before attempting to install the plugin.

## Installing the Artifactory Plugin

The Artifactory plugin is available from the Downloads section of the Flexera Customer Care website. Use the following steps to install the plugin.



### Task

**To install the Artifactory plugin, do the following:**

1. Download and extract the Artifactory plugin subfolder from the `CodeInsightversionPlugins.zip` file. For more information, see [Downloading Plugins](#).
2. Copy the following directory and files into the `<ARTIFACTORY_HOME>/etc/plugins` directory on the Artifactory server:
  - `libs` directory
  - `code-insight-scan-plugin.groovy`
  - `code-insight-scan.plugin.props`
3. Define the properties in the `code-insight-scan.plugin.props` file:

```
repoKeys=<REPOSITORY_PATH1>/,<REPOSITORY_PATH2>
codeinsight.server= http(s)://<HOST>:<PORT>/
codeinsight.auth.token=Bearer <JWT_TOKEN>
codeinsight.project.name= <PROJECT_NAME1>,<PROJECT_NAME2>
plugin.root.path=<./artifactory-pro-5.10.2/etc/plugins>
plugin.project.description= will be set by plugin, can be left blank
isScanCronJobEnabled=disabled
isPluginEnabled=enabled
cronJobTime=1 * * * * ?
artifactory_url= http(s)://<HOST>:<PORT>/ARTIFACTORY/
```
4. Determine if you want to execute the scan with a cron job or by calling REST API:
  - To execute a scan with a cron job, see [Scanning an Artifactory Repository Using a Cron Job](#).
  - To execute a scan by calling REST API, see [Scanning an Artifactory Repository Using REST API](#).

## Scanning an Artifactory Repository Using a Cron Job

You can use the following procedure to schedule an Artifactory repository scan to run periodically.

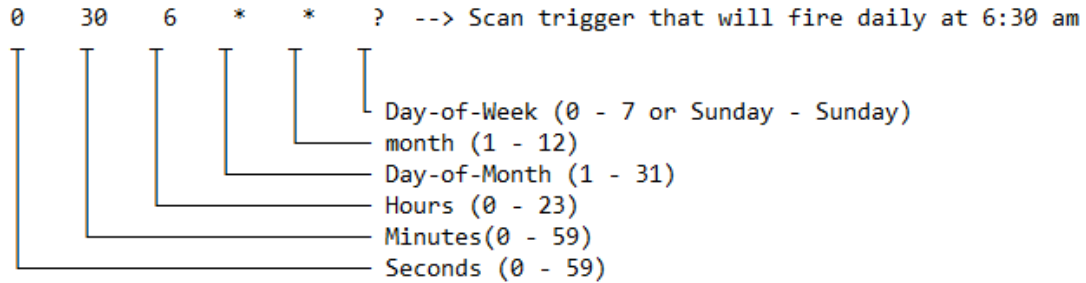




#### Task

To execute an Artifactory scan using a cron job, do the following:

1. Open the `code-insight-scan.plugin.props` file.
2. Modify the property `isScanCronJobEnabled=disabled` to `isScanCronJobEnabled=enabled`.
3. Set the `cronJobTime` property to schedule the scan. Use the following diagram and the example it provides to help you set the property.



4. Restart the Artifactory server.

## Scanning an Artifactory Repository Using REST API

You can call REST API to scan all Artifactory repositories listed in the `code-insight-scan.plugin.props` file or to scan a specific repository instead. The following topics describe how to scan repositories using REST API:

- [Requirements When Using REST API to Scan Artifactory Repositories](#)
- [Scanning All Artifactory Repositories](#)
- [Scanning a Specific Artifactory Repository](#)
- [Reloading the Artifactory Plugin](#)

## Requirements When Using REST API to Scan Artifactory Repositories

The following lists the requirements for using REST APIs to scan Artifactory repositories.

### Prerequisite for Scanning Repositories

As prerequisite for using REST API to scan Artifactory repositories, ensure that the properties in `code-insight-scan.plugin.props` are properly defined according to the instructions in [Installing the Artifactory Plugin](#) and according to any specific instructions listed in the procedures.

### Required Option When Using the “https” Protocol

The REST API calls used in the next sections use the `http` protocol. To use the `https` protocol instead, be sure to include the option `-k` in the call:

```
curl -X POST -u<USER_NAME>:<PASSWORD> -k "https://<ARTIFACTORY_HOST>:8081/artifactory/api/plugins/execute/CodeInsightScan"
```

## Scanning All Artifactory Repositories

The following command scans all repositories listed in the `code-insight-scan.plugin.props` file.



---

**Task**      *To scan all repositories, do the following:*

Use the following API call to scan all repositories:

```
curl -X POST -u<USER_NAME>:<PASSWORD> "http://<ARTIFACTORY_HOST>:8081/artifactory/api/plugins/execute/CodeInsightScan"
```

## Scanning a Specific Artifactory Repository

The following procedure scans a specific repository.



---

**Task**      *To scan a specific repository, do the following:*

1. Ensure that the following properties are also defined in the `code-insight-scan.plugin.props` file: `codeinsight.server`, `codeinsight.auth.token`, `plugin.root.path`, `isPluginEnabled`, and `artifactory_url`.
2. Use the following API call to scan the repository:

```
curl -X POST -u<USER_NAME>:<PASSWORD> "http://<ARTIFACTORY_HOST>:8081/artifactory/api/plugins/execute/CodeInsightSingleScan?params=repoKey=<REPOSITORY_NAME>%7cproject=<FNCCI_PROJECT_NAME>"
```

## Reloading the Artifactory Plugin

If you have downloaded an updated version of the Code Insight plugin for Artifactory, you can use this REST API call to reload the plugin before running a scan:

```
curl -X POST --u<USER_NAME>:<PASSWORD> http://localhost:8081/artifactory/api/plugins/reload
```

## Scan Results

When the scan completes, inventory is created in the corresponding FlexNet Code Insight project. The **Scan Status** section on the **Summary** tab for the project provides information about the scan.

Similarly in Artifactory, information about the scan, such as the Code Insight project name, the scan status, and a link to the Code Insight project inventory are provided for each repository scanned. For more information about using plugins in Artifactory, see the following site:

<https://www.jfrog.com/confluence/display/RTF/User+Plugins>

# Plugins for Container Platforms

Currently, FlexNet Code Insight support for scan integration with a container platforms includes the [Docker Images Plugin](#).

## Docker Images Plugin

Docker is a tool that packages applications and their dependencies into containers, which are comprised of static images. These images are themselves comprised of layers. Code Insight Docker Images scan agent enables the scanning of Docker images on a Docker server and sends the results as inventory to the Code Insight server for review, management, and remediation.



**Note** • It is recommended that Docker images be scanned on a development, test, or staging server before being pushed to a production instance as part of the DevOps process flow.

The following topics describe how to install and launch the Docker Images plugin:

- [Prerequisites for the Docker Images Plugin](#)
- [Installing the Docker Images Plugin](#)
- [Launching the Docker Images Plugin](#)

## Prerequisites for the Docker Images Plugin

Before you install and configure the Docker Images plugin, ensure that the following prerequisites are met:

- The Docker server must be installed and configured properly in your environment. The Docker plugin can only be executed on a server that already has an authenticated connection to the Docker server.



**Note** • The Docker plugin issues Docker commands without prompting for credentials.

- All FlexNet Code Insight prerequisite tasks for plugins have been performed, as described in [Preparing to Use the Plugins](#).
- A minimum of 2GB of heap space is allocated on the Docker server, which must be configured with a 64-bit JRE to support that amount of heap space.
- The plugin sets the maximum JVM heap size to 4GB by default, a size sufficient for most images. However, if you are receiving errors due to heap size, you can increase this maximum by setting a higher `-Xmx` value in the last line of code in the `code-insight-docker-plugin.sh` file, located in the plugin folder. (You can also add a minimum heap size, using the `-Xms` option, if needed.) The following provides an example of how you might update this line to increase the maximum heap size:

```
java -Xmx10240m -Xms1024m -jar $DEBUG code-insight-docker-plugin.jar $*
```



**Note** • If you are using a 64-bit JVM, you can increase the heap size to the size you need. If you are using a 32-bit JVM, you are limited to 4GB.

## Installing the Docker Images Plugin

Use the following procedure to install the Docker Images plugin.



### Task

**To install Docker Images plugin, do the following:**

1. Extract the Docker Images plugin subfolder from the `CodeInsightversionPlugins.zip` file, and copy it the Docker server. For more information, see [Downloading Plugins](#).
2. Open the `code-insight.docker.props` file in a text editor:

```
//required
codeinsight.server=http://127.0.0.1:8888
codeinsight.auth.token=Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pb2IiInVzZXJJZCI6MSwiaWF0IjoxNTE4MjM0MDk4fQ.dHItJjJ2c89Dg5cVLvf
GR3fwJcR3yAlVE6k98dRZTdp3h6McDgv_PloVVE88eJ2GOG0tNDOnhU0ShDLUzdu3Pg
codeinsight.project.name=inv2
plugin.root.path=/Users/ranimathur/Work/Scratch/
//optional
plugin.project.name=plugin project name
plugin.project.description=plugin project description
plugin.path.prefix=$demo_workspace/
```
3. Edit the `code-insight.docker.props` file to specify the following information:
  - **codeinsight.server (required):** The URL path to the FlexNet Code Insight server.
  - **codeinsight.auth.token (required):** The JWT authentication token that you obtain from the FlexNet Code Insight server using the Bearer schema. See [Generating a JWT Authorization Token](#).
  - **codeinsight.project.name (required):** The name of the FlexNet Code Insight project.
  - **plugin.root.path (required):** The root path where the Docker plugin will be executing. This path must have writable privileges for the user executing the plugin.
  - **plugin.project.name (optional):** A descriptive name to the project being scanned, that may be different from the project name specified in the FlexNet Code Insight server. This text will appear on the **Summary** tab for the project in the FlexNet Code Insight user interface.
  - **plugin.project.description (optional):** A description of the project being scanned. This text will appear on the **Summary** tab for the project in the FlexNet Code Insight user interface.
  - **plugin.path.prefix (optional):** The path prefix of the image being scanned. This prefix will be used to reference the file paths of the codebase on the **Project Inventory** page of the FlexNet Code Insight GUI.

## Launching the Docker Images Plugin

Use the following procedure to launch the Docker Images plugin to scan an image.



**Note** • The Docker plugin must be launched whenever the Docker image is updated. The Docker plugin can be included in a script, so the image is scanned regularly.

**Task****To launch the Docker Images plugin, do the following:**

Issue the following command to launch the Docker plugin from the command line:

```
% code-insight-docker-plugin.sh -image <DOCKER_IMAGE_NAME>
```

The <DOCKER\_IMAGE\_NAME> is the name given to the image that FlexNet Code Insight is to scan.



**Note** • Only the downloaded Docker image is scanned.

As it runs, the Docker plugin does the following:

- Contacts the FlexNet Code Insight server to validate the connection and download a scanner.
- Issues the Docker commands to download the Docker image.
- Extracts the Docker image.
- Scans the extracted Docker image contents.

The plugin sends the inventory results to FlexNet Code Insight configured.

## Generic Scan Agent Plugin

The generic scan agent plugin is an example ready-to-use plugin that is available in the `codeinsight-generic-version` toolkit installed with the standard Code Insight plugins. It can run as a standalone scan agent plugin (as described in this section), enabling you to scan any file system instead of being restricted to specific Engineering systems as you are with the standard scan agent plugins. The plugin can also easily integrate with certain Engineering systems to perform scans as part of a build process, such a TeamCity or GitLab build process, or serve as a basis for developing your own scan agent plugin (see the [Developing Custom Plugins](#) chapter).

The following describe the basics of using the generic scan agent plugin as a standalone scan agent:

- [Prerequisites for the Generic Scan Agent Plugin](#)
- [Running the Generic Scan Agent Plugin](#)

## Prerequisites for the Generic Scan Agent Plugin

The following prerequisites are required to use the FlexNet Code Insight generic scan agent plugin:

- A minimum of 4 GB Java heap space required for scanning (allocated to the JVM under which the scan agent plugin will be executing).
- 64-bit Java Runtime Environment JRE 8 or later.
- FlexNet Code Insight Server.
- The Code Insight prerequisites described in [Preparing to Use the Plugins](#).

- Internet access (recommended but not required):
  - If Internet access is available, the scan agent will periodically download the latest security vulnerability definitions from the National Vulnerability Database (NVD).
  - If Internet access is not available, then the default signatures that were released with the latest version of FlexNet Code Insight will be used.

## Running the Generic Scan Agent Plugin

The generic scan agent plugin can scan any file system of your choice, without your being limited to a specific build system as you are with the standard scan agent plugins.

The scan returns the results back to Code Insight, where the discovered inventory items for your project can be reviewed automatically via policies or manually reviewed by various stakeholders. Security alerts with corresponding email notifications will be generated for any inventory items with new security vulnerabilities.



### Task

**To run the generic scan agent plugin, do the following:**

1. Download and extract the contents of the `CodeInsightversionPlugins.zip` file, as described in the previous section, [Downloading Plugins](#).
2. Locate the `code-insight-agent-sdk-generic/generic-plugin-binary` folder and copy it to your hard drive.
3. Run the following from a command line as a Java application:

```
java -Dflx.agent.logLevel=info -jar codeinsight-generic-<VERSION>.jar -server
"<FNCI_SERVER_HOSTNAME>:<PORT>/<FNCI_SERVER_PATH>" -token "Bearer <JWT_TOKEN>" -proj
"<FNCI_PROJECT_NAME>" -scandirs "</path/to/the/codebase/PROJECT>"
```

Replace the following variables with the appropriate information:

- `<VERSION>` with the version of the `.jar` file used to run the scan agent, as found in the `code-insight-agent-sdk-generic/generic-plugin-binary` folder.
- `<FNCI_SERVER_HOSTNAME>:<PORT>/<FNCI_SERVER_PATH>` with your Code Insight server URL (for example, <http://1.1.1.1:8888/codeinsight>)
- `<JWT_TOKEN>` with your JWT authorization token obtained from the Code Insight server (as described in [Generating a JWT Authorization Token](#)).
- `<FNCI_PROJECT_NAME>` with the Code Insight project you created to capture the inventory.
- `</path/to/the/codebase/PROJECT>` with the specific directories to scan.

Alternatively, you can set options and then run the application by executing `run_scan.bat` or `run_scan.sh`. These scripts are also located in the `generic-plugin-binary` folder.

# Developing Custom Plugins

While FlexNet Code Insight provides standard scan agent plugins that are ready to deploy for codebase scanning on remote Engineering servers, it also provides a Scan Agent toolkit that implements the Code Insight Scan Agent Framework, which enables you to write a custom scan agent plugin that integrates with your development ecosystem. The following sections provide guidance in creating a scan agent plugin:

- [Scan Agent Framework](#)
- [Downloading the Scan Agent Toolkit](#)
- [Contents of the Scan Agent Toolkit](#)
- [Writing a Custom Scan Agent Plugin](#)
- [Deploying a Custom Scan Agent Plugin](#)

## Scan Agent Framework

The Scan Agent Framework comprises a set of Java APIs that provide the ability to scan codebases at various phases of the development process on remote engineering systems, within the appropriate context. The Framework provides the backbone for all the processing that a user-created scan agent plugin requires.

The source code for a generic scan agent plugin is provided in the toolkit to demonstrate the API flow. By creating a plugin that takes advantage of the Scan Agent Framework, you can tailor FlexNet Code Insight's powerful scanning capabilities to your computing environment and incorporate them into your business process flow.

The following describes the Scan Agent Framework:

- [Features Provided by the Framework](#)
- [Available Classes and Methods in the Framework](#)
- [Property Settings](#)

## Features Provided by the Framework

The Scan Agent Framework provides the following functionality for the custom scan agent plugin:

- Tests the connection from your plugin to the FlexNet Code Insight server and provides error handling with error messages. These messages include the FlexNet Code Insight version, any invalid URLs passed, invalid user access tokens, and invalid project names.
- Passes environmental and system properties.
- Downloads and installs a remote scanner on the engineering system where the scan agent plugin is executed.
- Invokes the scan called for in the plugin, and sends the scan results back to FlexNet Code Insight.
- Processes and displays logging content to a system console in the scan agent plugin environment.
- Generates a verbose scanner log for further information and debugging of failed scans.
- Automatically uploads the output of the plugin to the FlexNet Code Insight server, where it is then available for inventory review, management, and security alerting via the Code Insight user interface.

## Available Classes and Methods in the Framework


The Scan Agent Framework provides the following classes and methods that can be used to build a scan plugin:

**Table 3-1** • Available Classes and Methods

Class	Description	Settings/Methods
<b>ExecutionContext</b>	Stores all the information needed for a scan to run and publish results to FlexNet Code Insight server. This class should be initialized only by calling the <code>ExecutionContext.getInstance(...)</code> method.	<b>Methods:</b>  <code>ExecutionContext getInstance(Properties props, PrintStream logger)</code>  Initializes ExecutionContext for the current scan.  <b>Parameters:</b>  <code>props</code> : Properties required to scan and publish results  <code>logger</code> : Output stream where all the useful logging will be sent; if logger is null, all logging is redirected to console output. For more information about parameters for this class, see <a href="#">Property Settings</a> .



Table 3-1 • Available Classes and Methods

Class	Description	Settings/Methods
<b>ScanExecutor</b>	<p>Executes the scan and the publish results request made by client plugins.</p>  <p><b>Note</b> • This class requires a valid <code>ExecutionContext</code> instance to be able to operate.</p>	<p><b>Methods:</b></p> <ul style="list-style-type: none"> <li>● <b><code>ScanExecutor(ExecutionContext executionContext)</code></b> A constructor.</li> <li>● <b><code>String testConnection()</code></b> Validates the user authorization token and the connection to the FlexNet Code Insight server.</li> </ul> <p><b>Valid returned strings:</b></p> <ul style="list-style-type: none"> <li>● Success</li> <li>● Server Not Found</li> <li>● Invalid Auth Token Found</li> <li>● User not authorized to access the project</li> </ul> <li>● <b><code>String scanCodebase(List&lt;String&gt; paths)</code></b> Updates the embedded scanner if updates are found, scans the paths provided as input, and publishes the results to the Code Insight server as provided for <code>ExecutionContext</code>.</li> <p><b>Parameters:</b></p> <p><b>paths:</b> list of absolute paths to be scanned</p> <p><b>Return:</b></p> <p>SUCCESS or FAILURE as a string. See the logger output stream for more details.</p>

## Property Settings

The following table lists property settings that you can update:

Table 3-2 • Property Settings

Property	Required/Optional?	Description
<b>codeinsight.server</b>	Required	Code Insight server URI (for example, <a href="http(s)://your.codeinsight.server:port/">http(s)://your.codeinsight.server:port/</a> ).
<b>codeinsight.auth.token</b>	Required	Authorized user token generated in the FlexNet Code Insight user interface.
<b>codeinsight.project.name</b>	Required	The FlexNet Code Insight project name where all the inventory information will be published.
<b>plugin.root.path</b>	Required	Local root path that can be replaced with path prefix.

Table 3-2 • Property Settings

Property	Required/Optional?	Description
<b>plugin.project.name</b>	Optional	Name of the plugin (for example, "Codeinsight Jenkins plugin").
<b>plugin.project.description</b>	Optional	Plugin's instance name/build number or tag can be sent to the Code Insight server.
<b>plugin.path.prefix</b>	Optional	Prefix to be added to file paths for display to the user. For example, it could be the URL for a Jenkins workspace.

## Downloading the Scan Agent Toolkit

Use the following procedure to download and extract the Scan Agent toolkit.



### Task

*To download the Scan Agent toolkit, do the following:*

1. Log into the Customer Community page of the Flexera website:  
<https://flexeracommunity.force.com/customer/>
2. Click **Downloads**.
3. Click the **Access** button under **FlexNet Code Insight**. The Product and License Center page appears.
4. Select **FlexNet Code Insight** from the **Your Downloads** list.
5. Select the version of FlexNet Code Insight from the list. The **Downloads** page appears.
6. Download the `CodeInsightversionPlugins.zip` file.
7. When the download finishes, extract the subfolder `code-insight-agent-sdk-generic-plugin`, which contains the toolkit.

Ensure that you extract the entire subfolder into your installation directory, so you have all necessary files to create the plugin.

## Contents of the Scan Agent Toolkit

The contents of the Scan Agent toolkit includes the following:

- **/readme.txt**: Instructions on how to use the SDK.
- **/lib**: All dependent jar files needed to run the plugin.
- **/generic-plugin-binary**: An example binary—the generic scan agent plugin—developed with the Scan Agent Framework.
- **/generic-plugin-source**: The source code for the example binary.
- **/generic-plugin-source/pom.xml+assembly.xml**: Maven build files used to compile and build the plugin.

# Writing a Custom Scan Agent Plugin

Writing a custom scan agent plugin involves the following the following tasks:

- **Task 1**— Review the example generic scan agent plugin for the code structure and build configuration.
- **Task 2**—Identify the Engineering system with which you will be integrating the scan agent. (This section and the next will use *Integration Server* as an example Engineering system).
- **Task 3**—Using the APIs provided by the Integration Server, write code to pull in all the codebase files that you want to scan into a single folder. This will be the folder path that you will be passing to the plugin.
- **Task 4**—Use the Scan Agent Framework APIs to connect to the FlexNet Code Insight server.
- **Task 5**—Scan the folder that contains the desired code base files on the Integration Server.



---

**Note** • *The plugin will typically execute on the same computer system as the Integration Server.*

# Deploying a Custom Scan Agent Plugin

Deployment of a custom scan agent plugin involves the following tasks:

- **Task 1**—Install and configure the FlexNet Code Insight server.
- **Task 2**—Ensure you adhere to the system prerequisites for the generic scan agent plugin. See [Prerequisites for the Generic Scan Agent Plugin](#).
- **Task 3**—Invoke the remote scans on the Integration Server, and review the results in the FlexNet Code Insight user interface.

