Legal Information

Book Name: FlexNet Code Insight 2020 R2 Plugins Guide
Part Number: FNCI-2020R2-PG00
Product Release Date: May 2020

Copyright Notice

Copyright © 2020 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.flexerasoftware.com/legal/intellectual-property.html. 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

1 FlexNet Code Insight 2020 R2 Plugins Guide ......................................................... 7
   About Scan Agent Plugins ................................................................................. 7
   Contents of this Book ....................................................................................... 8
   Product Support Resources ............................................................................. 8
   Contact Us ......................................................................................................... 9

2 Installing and Configuring Standard Plugins ....................................................... 11
   About Scan Agent Plugins ............................................................................. 12
   Overview of Available Plugins ..................................................................... 12
   Preparing to Use the Plugins ....................................................................... 13
   Providing an Authorization Token ............................................................... 13
   Downloading Plugins .................................................................................... 13
   Plugins for Integrated Development Environments (IDEs) ......................... 14
      Eclipse Plugin ............................................................................................... 14
      Prerequisites for the Eclipse Plugin .......................................................... 14
      Installing the Eclipse Plugin ..................................................................... 15
      Configuring the Eclipse Plugin ................................................................ 17
      Running a Scan within Your Environment .............................................. 18
      Uninstalling the Eclipse Plugin ............................................................... 19
      Visual Studio Plugin ................................................................................... 20
      Prerequisites for the Visual Studio Plugin ............................................... 20
      Installing the Visual Studio Plugin .......................................................... 20
      Configuring the Visual Studio Plugin ...................................................... 21
         Configuration Using Visual Studio IDE ............................................... 22
         Configuration Using MSBuild ................................................................ 24
      Executing a Scan ......................................................................................... 25
         Scan Execution Using Visual Studio IDE ........................................... 26
         Scan Execution Using MSBuild ............................................................ 27
## Contents

Disabling or Uninstalling the Visual Studio Plugin ................................................................. 27

### Plugins for Continuous Integration (CI) Tools ........................................................................ 28

<table>
<thead>
<tr>
<th>Tool</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure DevOps Extension</td>
<td>28</td>
</tr>
<tr>
<td>Prerequisites for the Azure DevOps Extension</td>
<td>28</td>
</tr>
<tr>
<td>Installing the Azure DevOps Extension</td>
<td>29</td>
</tr>
<tr>
<td>Adding a Scan Task to Your Azure DevOps Agent Job</td>
<td>29</td>
</tr>
<tr>
<td>Bamboo Plugin</td>
<td>31</td>
</tr>
<tr>
<td>Prerequisites for the Bamboo Plugin</td>
<td>32</td>
</tr>
<tr>
<td>Installing and Configuring the Bamboo Plugin</td>
<td>32</td>
</tr>
<tr>
<td>GitLab Plugin</td>
<td>34</td>
</tr>
<tr>
<td>Prerequisites for the GitLab Plugin</td>
<td>34</td>
</tr>
<tr>
<td>Installing the Generic Scan Agent on GitLab Runner</td>
<td>34</td>
</tr>
<tr>
<td>Configuring the CI/CD Pipeline to Run a Scan</td>
<td>35</td>
</tr>
<tr>
<td>Executing the Build</td>
<td>36</td>
</tr>
<tr>
<td>Jenkins Plugin</td>
<td>36</td>
</tr>
<tr>
<td>Prerequisites for the Jenkins Plugin</td>
<td>36</td>
</tr>
<tr>
<td>Setting Heap Size for the Jenkins Plugin</td>
<td>36</td>
</tr>
<tr>
<td>Setting Up the Code Insight Jenkins Plugin</td>
<td>37</td>
</tr>
<tr>
<td>Support for the Jenkins Pipeline</td>
<td>37</td>
</tr>
<tr>
<td>Providing the Pipeline Script for the Scan Step</td>
<td>38</td>
</tr>
<tr>
<td>Pipeline Code Examples for Running the Scan</td>
<td>39</td>
</tr>
<tr>
<td>Scan Scheduler Plugin for Jenkins</td>
<td>40</td>
</tr>
<tr>
<td>TeamCity Plugin</td>
<td>41</td>
</tr>
<tr>
<td>Prerequisites for TeamCity Plugin</td>
<td>41</td>
</tr>
<tr>
<td>Installing the Generic Scan Agent on the Team City Build Agent</td>
<td>41</td>
</tr>
<tr>
<td>Configuring a Build to Run a Code Insight Scan</td>
<td>42</td>
</tr>
<tr>
<td>Executing the Build</td>
<td>43</td>
</tr>
</tbody>
</table>

### Plugins for Package Managers and Build Tools ................................................................. 44

<table>
<thead>
<tr>
<th>Tool</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Ant Plugin</td>
<td>44</td>
</tr>
<tr>
<td>Prerequisites for the Apache Ant Plugin</td>
<td>44</td>
</tr>
<tr>
<td>Configuring the Apache Ant Plugin</td>
<td>44</td>
</tr>
<tr>
<td>Executing the Scan</td>
<td>45</td>
</tr>
<tr>
<td>Gradle Plugin</td>
<td>46</td>
</tr>
<tr>
<td>Prerequisites for the Gradle Plugin</td>
<td>46</td>
</tr>
<tr>
<td>Installing and Configuring the Gradle Plugin</td>
<td>46</td>
</tr>
<tr>
<td>Maven Plugin</td>
<td>48</td>
</tr>
<tr>
<td>More About the Maven Plugin</td>
<td>48</td>
</tr>
<tr>
<td>Prerequisites for the Maven Plugin</td>
<td>48</td>
</tr>
<tr>
<td>Installing and Configuring the Maven Plugin</td>
<td>48</td>
</tr>
<tr>
<td>Cleaning the Application Project</td>
<td>50</td>
</tr>
<tr>
<td>Running the Maven Goal for the Scan</td>
<td>51</td>
</tr>
</tbody>
</table>

### Plugins for Binary Repositories ...................................................................................... 51

<table>
<thead>
<tr>
<th>Tool</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFrog Artifactory Plugin</td>
<td>51</td>
</tr>
<tr>
<td>Prerequisites for the Artifactory Plugin</td>
<td>51</td>
</tr>
<tr>
<td>Installing the Artifactory Plugin</td>
<td>52</td>
</tr>
<tr>
<td>Scanning an Artifactory Repository Using a Cron Job</td>
<td>52</td>
</tr>
</tbody>
</table>
## Scanning an Artifactory Repository 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

## Scan Results

## Plugins for Container Platforms

- Docker Images Plugin
  - Prerequisites for the Docker Images Plugin
  - Installing the Docker Images Plugin
  - Launching the Docker Images Plugin

## Generic Scan Agent Plugin

- Prerequisites for the Generic Scan Agent Plugin
- Running the Generic Scan Agent Plugin
- Enabling the Generic Scan Agent Plugin to Detect Transitive Dependencies

## Developing Custom Plugins

- Scan Agent Framework
  - Features Provided by the Framework
  - Available Classes and Methods in the Framework
  - Property Settings
- Downloading the Scan Agent Toolkit
- Contents of the Scan Agent Toolkit
- Writing a Custom Scan Agent Plugin
- Deploying a Custom Scan Agent Plugin
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) directly in your development environment to scan your product source files and post-build artifacts for OSS and third-party components.

This chapter covers the following information:

- About Scan Agent Plugins
- Contents of this Book
- Product Support Resources
- Contact Us

About Scan Agent Plugins

FlexNet Code Insight supports a scan agent plugin, which is installed directly in your development environment to perform scans on your product’s source files and built artifacts as part of your software development process. This type of scan is an alternative to the standard scan, which is performed source codebase files that are uploaded to Code Insight. (Scans on codebases uploaded to the Scan Server are described in FlexNet Code Insight User Guide.)

The scan agent plugin is configured to scan a specific set of files within the context of an Engineering application (such as an IDE, artifact-repository, CI, build, testing, installation, or source-management application). Once configured, the plugin can be invoked to run a scan as part of the build process. The results are sent back to Code Insight as published inventory awaiting review, management, and remediation. Just 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. 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 ready for immediate deployment. 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 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**

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

Product Support Resources

The following resources are available to assist you with using this product:

- Flexera Product Documentation
- Flexera Community
- Flexera Learning Center
- Flexera Support

**Flexera Product Documentation**

You can find documentation for all Flexera products on the Flexera HelpNet site:

https://helpnet.flexerasoftware.com

**Flexera Community**

On the Flexera Community site, you can quickly find answers to your questions by searching content from other customers, product experts, and thought leaders. You can also post questions on discussion forums for experts to answer. For each of Flexera’s product solutions, you can access forums, blog posts, and knowledge base articles.

https://community.flexera.com
Flexera Learning Center

Flexera offers a variety of training courses—both instructor-led and online—to help you understand how to quickly get the most out of your Flexera products. The Flexera Learning Center offers free, self-guided, online training classes. You can also choose to participate in structured classroom training delivered as public classes. You can find a complete list of both online content and public instructor-led training in the Learning Center.

https://learn.flexera.com

Flexera Support

For customers who have purchased a maintenance contract for their product(s), you can submit a support case or check the status of an existing case by making selections on the Get Support menu of the Flexera Community.

https://community.flexera.com

Contact Us

Flexera is headquartered in Itasca, Illinois, and has offices worldwide. To contact us or to learn more about our products, visit our website at:

http://www.flexera.com

You can also follow us on social media:

- Twitter
- Facebook
- LinkedIn
- YouTube
- Instagram
Installing and Configuring Standard Plugins

FlexNet Code Insight supports scan agent plugins, which are installed directly in development environments to perform scans on pre-build source files or post-build artifacts as part of the software development process. These remote plugin scans provide an alternative to scans performed on codebases that are uploaded to the Code Insight, as described in the FlexNet Code Insight User Guide.

The following sections discuss the download, 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
- Providing an 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>
<thead>
<tr>
<th>Build Environment</th>
<th>Code Insight Plugin</th>
<th>Performs automated scanning of...</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEs</td>
<td>Eclipse Plugin</td>
<td>An Eclipse workspace in the Eclipse IDE environment.</td>
</tr>
<tr>
<td>Ci Tools</td>
<td>Azure DevOps</td>
<td>An Azure DevOps workspace as part of the build process.</td>
</tr>
<tr>
<td></td>
<td>Extension</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bamboo Plugin</td>
<td>A Bamboo workspace as part of the build process (on Local Agents only)</td>
</tr>
<tr>
<td></td>
<td>GitLab Plugin</td>
<td>GitLab projects as part of the build process.</td>
</tr>
<tr>
<td></td>
<td>Jenkins Plugin</td>
<td>A Jenkins workspace as part of the build process.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>TeamCity Plugin</td>
<td>TeamCity projects as part of the build process.</td>
</tr>
<tr>
<td>Package Manager</td>
<td>Apache Ant Plugin</td>
<td>Apache Ant as part of the build process.</td>
</tr>
<tr>
<td>and Build Tools</td>
<td>Gradle Plugin</td>
<td>Gradle projects as part of the build process.</td>
</tr>
<tr>
<td></td>
<td>Maven Plugin</td>
<td>Maven projects as part of the build process.</td>
</tr>
<tr>
<td>Binary Repositories</td>
<td>JFrog Artifactory</td>
<td>Artifactory repositories to identify non-compliant artifacts.</td>
</tr>
<tr>
<td></td>
<td>Pluginy</td>
<td></td>
</tr>
<tr>
<td>Container Platforms</td>
<td>Docker Images</td>
<td>Docker images on a Docker server.</td>
</tr>
<tr>
<td></td>
<td>Plugin</td>
<td></td>
</tr>
</tbody>
</table>

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).
Chapter 2  Installing and Configuring Standard Plugins

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 Providing an Authorization Token for more information.

• 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.

Providing an Authorization Token

FlexNet Code Insight uses a JSON Web Token (JWT) to authorize user access to the Code Insight public REST interface. Several of the scan agent plugins make use of the REST APIs and thus require a JWT. For information about obtaining this authorization token, see “Managing Authorization Tokens” in the “Using FlexNet Code Insight” chapter in the FlexNet Code Insight User Guide.

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 the Flexera Customer Community page. The following procedure assumes you have a login and password to access the Customer Community page at https://flexeracommunity.force.com/customer/.

<table>
<thead>
<tr>
<th>Task</th>
<th>To download the plugin zip file, do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Log into the Flexera Customer Community page:</td>
</tr>
<tr>
<td></td>
<td><a href="https://flexeracommunity.force.com/customer/">https://flexeracommunity.force.com/customer/</a></td>
</tr>
<tr>
<td>2.</td>
<td>Click Downloads.</td>
</tr>
<tr>
<td>3.</td>
<td>Click the Access button under FlexNet Code Insight. The Product and License Center page appears.</td>
</tr>
<tr>
<td>4.</td>
<td>Select FlexNet Code Insight from the Your Downloads list.</td>
</tr>
<tr>
<td>5.</td>
<td>Select the version of FlexNet Code Insight from the list. The Downloads page appears.</td>
</tr>
</tbody>
</table>
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 environment (IDE) includes the following:

- Eclipse Plugin
- Visual Studio 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.
3. On the **Available Software** window, click the **Add** to display the **Add Repository** popup:

![Add Repository popup](image)

4. On the popup, click **Archive**.

5. 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:
6. 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.

![Install Details Window]

7. Click Next to display the Review Licenses window.

8. Accept the license agreement terms, and click Finish.

9. 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:

1. In Eclipse, select the project whose codebase you want to scan, right-click the project entry, and select Properties.

2. 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server URL</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Project Name</strong></td>
<td>The name of the inventory-only project that was created in the Code Insight user interface (for example, <strong>ScanProject2_eclipse</strong>).</td>
</tr>
<tr>
<td><strong>Token</strong></td>
<td>The JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an Authorization Token.</td>
</tr>
<tr>
<td><strong>Scan Directory</strong></td>
<td>The folder containing the code to scan (for example, C:\Users\user1\eclipse-workspace\project2_code).</td>
</tr>
</tbody>
</table>

4. 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.

5. 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.
Chapter 2  Installing and Configuring Standard Plugins

Plugins for Integrated Development Environments (IDEs)

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:

![FlexNet Code Insight Scan Window](image)

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 opens to the **Project Inventory** tab for the inventory-only project created for the scan. From here you can review, manage, and remediate the inventory resulting from the scan. For further instructions, refer to “Reviewing Published Inventory” 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.
Visual Studio Plugin

The FlexNet Code Insight plugin for Visual Studio (called the Visual Studio plugin) enables development teams perform Code Insight scans within their Microsoft Visual Studio 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 Visual Studio plugin and configure it for your Visual Studio solution, as described in the following topics. Plugin configuration and scan execution can be performed through either the Visual Studio IDE interface or MSBuild.

- Prerequisites for the Visual Studio Plugin
- Installing the Visual Studio Plugin
- Configuring the Visual Studio Plugin
- Executing a Scan
- Disabling or Uninstalling the Visual Studio Plugin

Prerequisites for the Visual Studio Plugin

Before you can install and configure the Visual Studio plugin, perform the required tasks described in Preparing to Use the Plugins. Note that one of these required tasks is to create an inventory-only project on the Code Insight server in which to store scan results for analysis and review in Code Insight. If you prefer, you can have the configuration process for the Visual Studio plugin create this project for you, as described later in Configuring the Visual Studio Plugin.

**Note** • Microsoft Visual Studio Express does not support the Visual Studio plugin.

Installing the Visual Studio Plugin

Use Visual Studio IDE to install the Visual Studio plugin extension

**Task** To install the Visual Studio plugin, do the following:

1. In Visual Studio IDE, select **Tools | Extensions and Updates**.
2. In the **Extensions and Updates** tree, search for the **FlexNet Code Insight for Visual Studio** extension in the **Online | Visual Studio Marketplace** category.
3. When you locate the extension, click **Download** to download and automatically launch the plugin installer. You are prompted to select the Visual Studio version to which you are installing the plugin.

4. Select the appropriate Visual Studio version, and click **Install**.

5. When the installation completes, restart Visual Studio IDE to apply the extension.

**Configuring the Visual Studio Plugin**

After the plugin is installed, it must be configured in Visual Studio to enable codebase scans for a specific solution. If you have not already created a Code Insight project in which to store the scan results on the Code Insight server, the configuration process can create this project for you.
You can configure the plugin either in Visual Studio IDE or through the MSBuild command interface:

- Configuration Using Visual Studio IDE
- Configuration Using MSBuild

Configuration Using Visual Studio IDE

During the Visual Studio plugin installation, the FlexNet Code Insight Properties icon is added to the Solution Explorer toolbar in Visual Studio IDE. This icon provides access to the settings needed to configure the plugin at the solution level.

The following steps describe this configuration process.

**Task**

*To use the Visual Studio IDE interface to configure the Visual Studio plugin, do the following:*

1. In Visual Studio IDE, open the Visual Studio solution that you intend to scan.

2. In the Solution Explorer toolbar, click the FlexNet Code Insight Properties icon.

The FlexNet Code Insight Properties dialog is displayed.
3. Complete the following fields on the dialog to configure the Visual Studio plugin for the current solution:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlexNet Code Insight Server</td>
<td>Enter the URL for the Code Insight core server in the format http://&lt;SERVERHOSTNAME&gt;:&lt;PORT&gt;/codeinsight/ (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.</td>
</tr>
<tr>
<td>Authentication Token</td>
<td>Provide the JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. You generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an Authorization Token.</td>
</tr>
<tr>
<td>FlexNet Code Insight Project</td>
<td>Enter the name of the Code Insight inventory-only project that already exists or that you want this configuration process to create on the Code Insight server to store scan results. If you want the configuration process to create the specified Code Insight project, select Create New Project (see the next table entry). If the specified project already exists on the Code Insight server, do not select Create New Project or Create Private Project (see the next table entries).</td>
</tr>
<tr>
<td>Create New Project</td>
<td>Select this option if you want the configuration process to create an inventory-only project in which to store scan results on the Code Insight server. The project is the name specified for FlexNet Code Insight Project. The Project Owner is the user who generated the JWT provided for Authentication Token. If you want this new project to be public—that is, viewable by all Code Insight users—leave the next option, Create Private Project, unselected. Otherwise, select Create Private Project (described next).</td>
</tr>
</tbody>
</table>
Chapter 2 Installing and Configuring Standard Plugins
Plugins for Integrated Development Environments (IDEs)

4. Click OK to save the plugin configuration.

Configuration Using MSBuild

The following steps describe how to use MSBuild to configure the Visual Studio plugin once it is installed.

Task

To use the Visual Studio IDE interface to configure the Visual Studio plugin, do the following:

1. Launch Visual Studio IDE in Run As Administrator mode to enable the Visual Studio plugin for MSBuild and create a project/solution. You need to perform this step only once.

2. Copy the template configuration file flexnetcodeinsight_scan_settings.ini from \LOCAL_APP_DATA\Local\Microsoft\VisualStudio to the Visual Studio solution folder you want to scan.

   The following is an example solution folder to which to copy the file: C:\Users\jsmith\Documents\Visual Studio 2015\Projects\MyProject.

3. In a text editor, open the flexnetcodeinsight_scan_settings.ini that you copied to the solution folder, and provide the following values in the Settings section:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlexNetCodeInsight Server</td>
<td>Provide the URL for the Code Insight core server in the format http://&lt;SERVERHOSTNAME&gt;:&lt;PORT&gt;/codeinsight/ (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.</td>
</tr>
</tbody>
</table>
4. Save the file changes.

**Executing a Scan**

Once the Visual Studio plugin has been properly configured, you can manually invoke a scan on your solution codebase whenever needed. Trigger the scan from either Visual Studio IDE or through MSBuild commands:

- **Scan Execution Using Visual Studio IDE**
- **Scan Execution Using MSBuild**
The Visual Studio plugin can also be configured to trigger a scan automatically at the end of each build or rebuild. (See Configuring the Visual Studio Plugin for configuration details.) When the scan completes, you can click the URL at the end of the build output to connect to Code Insight. You are opened to the Project Inventory tab, where you can review and remediate the project inventory resulting from the scan. Refer to “Reviewing Published Inventory” in the “Using FlexNet Code Insight” chapter in the FlexNet Code Insight User Guide.

Scan Execution Using Visual Studio IDE

Use the following procedure to manually invoke a Code Insight scan on your solution codebase in Visual Studio IDE.

Task  To run a scan using Visual Studio IDE, do the following:

1. In the Solution Explorer in your Visual Studio IDE, navigate to the solution you want to scan.
2. Right-click the solution entry, and select FlexNet Code Insight Scan to start the scan.
3. When the scan completes, click the URL at the end of the build output to connect to Code Insight. You are opened to the Project Inventory tab for the Code Insight project created for the scan. From here you can review, manage, and remediate the inventory resulting from the scan. For further instructions, refer to “Reviewing Published Inventory” in the “Using FlexNet Code Insight” chapter in the FlexNet Code Insight User Guide.
Scan Execution Using MSBuild

The following procedure uses MSBuild commands to manually invoke a Code Insight scan on your solution codebase.

Task To run a scan using MSBuild, do the following:

1. Open a command-line prompt as an administrator, navigate to the MSBuild directory.

   Note that these directories might vary based on your Visual Studio version:

<table>
<thead>
<tr>
<th>Visual Studio Version</th>
<th>MSBuild Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS2017</td>
<td>c:\Program Files (x86)\Microsoft Visual Studio\2017\INSTALLED_EDITION\MSBuild\15.0\Bin</td>
</tr>
<tr>
<td>VS2012, VS2015</td>
<td>C:\Windows\Microsoft.NET\Framework&lt;FRAMEWORK_VERSION&gt;\MSBuild</td>
</tr>
<tr>
<td>VS2015</td>
<td>C:\Program Files (x86)\MSBuild\14.0\Bin</td>
</tr>
</tbody>
</table>

2. Enter the following command:

   MSBuild.exe "<PATH_OF_SOLUTION_FILE>" /p:FlexNetCodeInsightScan=true

   The command uses these parameters:
   - `<PATH_OF_SOLUTION_FILE>`—The absolute path of the solution directory you are scanning.
   - `FlexNetCodeInsightScan`—The parameter indicating that you want to run a Code Insight scan. Set this parameter to `true`. If you omit this option or set it to `false`, no scan is run.
   - `FlexNetCodeInsightConfig`—(Optional) The absolute path of the .ini configuration file if it does not reside in the solution directory specified for `<PATH_OF_SOLUTION_FILE>`. If you provide a value for this option, use the following command syntax:

     MSBuild.exe "<PATH_OF_SOLUTION_FILE>" /p:FlexNetCodeInsightScan=true; FlexNetCodeInsightConfig=<ABSOLUTE_PATH_TO_INI_CONFIG_FILE>

3. When the scan completes, click the URL at the end of the build output to connect to Code Insight. You are opened to the Project Inventory tab for the Code Insight project created for the scan. From here you can review, manage, and remediate the inventory resulting from the scan. For further instructions, refer to “Reviewing Published Inventory” in the “Using FlexNet Code Insight” chapter in the FlexNet Code Insight User Guide.

Disabling or Uninstalling the Visual Studio Plugin

Use the following procedure to disable or uninstall the Visual Studio plugin.

Task To uninstall the Visual Studio plugin, do the following:

1. Open Visual Studio IDE, and select Tools | Extensions and Updates.

2. Select the Installed category in the left pane.

3. From the list of applications, select FlexNet Code Insight Scan Plugin, and click Disable or Uninstall.
Chapter 2  Installing and Configuring Standard Plugins

Plugins for Continuous Integration (CI) Tools

4. Restart Visual Studio IDE to verify that the plugin has been disabled or removed from the list of applications.

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.

<table>
<thead>
<tr>
<th>Task</th>
<th>To obtain and install the extension, do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Open the Visual Studio Marketplace:</td>
</tr>
<tr>
<td>2.</td>
<td>In the Azure DevOps section, search for the FlexNet Code Insight Scan extension.</td>
</tr>
<tr>
<td>3.</td>
<td>Download and install this extension into Azure DevOps.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Task</th>
<th>To add a scan task to your DevOps agent job, do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Create a build pipeline for your Azure DevOps project.</td>
</tr>
<tr>
<td>2.</td>
<td>Locate the FlexNet Code Insight Scan task under the Builds section in the task catalog.</td>
</tr>
</tbody>
</table>
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:
The following describes the task properties:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlexNet Code Insight Server</td>
<td>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.</td>
</tr>
<tr>
<td>Authorization Token</td>
<td>The JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an Authorization Token.</td>
</tr>
<tr>
<td>FlexNet Code Insight Project Name</td>
<td>The name of the inventory-only project that was created in the Code Insight user interface (for example, ScanProject2_AzureDevOps).</td>
</tr>
<tr>
<td>Folder(s) to Scan</td>
<td>The folder containing the code to scan. Typically, you would use $(build.artifactstagingdirectory), which is the location where the build output is staged during the build process.</td>
</tr>
</tbody>
</table>

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:

- All FlexNet Code Insight prerequisite tasks for plugins have been performed, as described in Preparing to Use the Plugins.
- Bamboo 5.2 or higher is installed and configured as explained in the Bamboo installation documentation.
- The Bamboo server instance on which you are running the plugin can be a Local Agent or Remote Agent.
- Maximum heap memory size is set to 4 GB for the Bamboo server (Local or Remote Agent, wherever the plugin is running). This size can be configured using the following property in wrapper.conf:
  \n  wrapper.java.maxmemory=4096
  
  Note that this is the recommended size value. However, heap size is relative to the size of the codebase. A large codebase requires this value to be increased 6GB or 8 GB.
- Event details for scans run on a Local Agent are logged to the \(<BAMBOO_HOME>/logs/atlassian-bamboo.log\) file. On a Bamboo Remote Agent, the events are logged to the \(<BAMBOO_REMOTE_DIR>/logs/atlassian-bamboo.log\) file.

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.

<table>
<thead>
<tr>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>To install and configure the Bamboo plugin, do the following:</td>
</tr>
</tbody>
</table>

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 by creating the plan, adding a job, and then adding a FlexNet Code Insight Scan task. To create the task, access the FlexNet Code Insight Scan Task Configuration window.
7. Enter the following information in the FlexNet CodeInsight Scan Task Configuration window:

- **Task description**—A label for this scan task.
- **Disable this task**—The option to disable or enable the scan task as needed.
- **Server URL**—The URL for the Code Insight core server (for example, `http://fnciserver.myorg.org:8888/codeinsight/`).
- **Authentication Token**—The JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an Authorization Token.
- **Project Name**—The name of the inventory-only project that you created in the FlexNet Code Insight to associate with this scan task.
- **Folder(s) to scan**—The one or more folders to scan.

If you want to scan the entire working folder, leave this field blank. However, to scan specific folders in the working directory, list the path for each folder as relative to the working folder, using commas to separate multiple folders.

For example, the working directory `opt/atlassian/workingDirectory/project` has the following source sub-folders:

```
/opt/atlassian/workingDirectory/project/source1
/opt/atlassian/workingDirectory/project/source2
/opt/atlassian/workingDirectory/project/source3
/opt/atlassian/workingDirectory/project/source4/source4a
```

- To scan the source1 folder only, enter `source1`. 
To scan both source1 and source2, enter the following:

```
source1, source2
```

To scan the source4a folder, enter `source4/source4a`.

8. Click **Save**. If the **Server URL** and **Token** values 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.


8. **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/](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

Use these instructions to install the generic scan agent plugin on the GitLab runner (installed on Windows).

<table>
<thead>
<tr>
<th>Task</th>
<th>To install the generic scan agent plugin on the GitLab runner, do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Download and extract the contents of the <code>CodeInsightversionPlugins.zip</code> file, as described in the previous section, <strong>Downloading Plugins</strong>.</td>
</tr>
<tr>
<td>2.</td>
<td>Locate the <code>code-insight-agent-sdk-generic-plugin/generic-plugin-binary</code> folder, and copy it to the GitLab runner.</td>
</tr>
</tbody>
</table>
3. 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
     ```

4. If you want the generic scan agent plugin to detect transitive dependencies during scans, follow the procedure described in Enabling the Generic Scan Agent Plugin to Detect Transitive Dependencies to configure this capability.

### 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:

```yaml
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:

- **<FNCI SERVER>**—The URL of the Code Insight Core Server (for example, `http://1.1.1.1:8888/codeinsight`).

- **<AUTH TOKEN>**—Your JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an Authorization Token.

- **<FNCI_PROJECT NAME>**—The inventory-only project you created in Code Insight to capture the inventory.

- **<tag for your GitLab-Runner>**—The tag for 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.
Chapter 2  Installing and Configuring Standard Plugins
Plugins for Continuous Integration (CI) Tools

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:

![Pipeline Jobs](image)

**Note** • 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.

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

Use this procedure to set the heap size in a Windows environment.

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

Use this procedure to set the heap size in a Linux environment.

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 might need 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.
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**—The JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an 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 scans 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) 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:

```groovy
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.

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

def checkout1(){
    dir("project-1"){
```
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.
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 — The JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an Authorization Token.
   - 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 Team City Build 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](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 Team City Build Agent).

**Installing the Generic Scan Agent on the Team City Build Agent**

Use these instructions to install the generic scan agent plugin on the Team City build agent.
To install the generic scan agent plugin on the Team City build agent, 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-plugin/generic-plugin-binary folder, and copy it to the TeamCity build agent.

3. 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
     ```

4. If you want the generic scan agent plugin to detect transitive dependencies during scans, follow the procedure described in Enabling the Generic Scan Agent Plugin to Detect Transitive Dependencies to configure this capability.

Configuring a Build to Run a Code Insight Scan

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

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.

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>** — The name of the inventory-only project you created in Code Insight to capture the inventory.
- **<FNCI_SERVER>** — The URL of the Code Insight Core Server (for example, `http://1.1.1.1:8888/codeinsight`).
- **<JWT_TOKEN>** — Your JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it here. For more information, see [Providing an Authorization Token](#)
- **<SCAN_DIR>** — The directory that you want to scan.

![Build Step](image)

When complete, your build configuration should look like this:

![Build Steps](image)

**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 this in your TeamCity build queue:
### Note
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.

## 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.

<table>
<thead>
<tr>
<th><strong>Task</strong></th>
<th><strong>To configure the plugin, do the following:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Extract the Ant plugin from the CodeInsight\texttt{version}Plugins.zip file. See Downloading Plugins.</td>
</tr>
<tr>
<td>2.</td>
<td>Configure %ANT_HOME% and add %ANT_HOME%/bin to the path variable.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Task</th>
<th>To execute the scan, do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Run the following command:</td>
</tr>
<tr>
<td></td>
<td><code>ant &lt;targetname&gt;</code></td>
</tr>
<tr>
<td></td>
<td>For example, you might enter:</td>
</tr>
<tr>
<td></td>
<td><code>ant compile</code></td>
</tr>
<tr>
<td>2.</td>
<td>To execute the scan along with any target of the build lifecycle, apply the plugin inside the target in the <code>build.xml</code> of the Ant application as follows:</td>
</tr>
</tbody>
</table>
|      | `<taskdef name="codeinsightantplugin" classname="com.ant.plugin.CodeInsightAntPlugIn" classpath="" classpathref="" />
|      | `<codeinsightantplugin fnceServer="<SERVER_URL>" fnceauthtoken="<BEARER_SERVER_TOKEN_VALUE>" fnceprojectname="<CODE_INSIGHT_PROJECT_NAME>" pluginRootPath="<PLUGIN_ROOT_PATH>" pluginProjectName="<APPLICATION_PROJECT_TO_SCAN>" pluginDescription="<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 cannot 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">
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 review, management, 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 1ib 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:

2. Use these steps to add all the dependent jars in the code-insight-gradle-plugin to the application class path:
   a. Create a folder named dependent-jars within the application project.
b. Copy all jar files into that folder.

c. Add the following configuration in build.gradle so that the jars are available to the classpath:

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

3. If the Java plugin is not already applied in the build.gradle script, do so by adding the appropriate configuration at the beginning of the script:

- For a single module project, add the following:
  ```gradle
  apply plugin: 'java'
  ```
- For a multi-modular project:
  ```gradle
  allprojects {
    apply plugin: 'java'
  }
  ```

4. Apply the Gradle plugin in the build.gradle file:

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

```gradle
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 scan settings used to apply the plugin:

- **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 JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it here. Be sure to include the command “Bearer” followed by the token value, as in the example:
  ```text
  Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsInVzZXJJZCI6MSwia
  ```
  For more information about generating this token, see Providing an 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.

5. 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.) You can always check your Maven installation by running `mvn -v`.
- 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.
To install and configure the Code Insight Maven plugin, do the following:

1. From the CodeInsight

   versionPlugins.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:

   ```
   
   ```

   Note the following variables:
   - `${PROJECT_DIRECTORY}` is your application project directory (or the local directory to which you extracted the plugin).
   - `<PLUGIN_VERSION>` is the latest version of the code-insight-maven-scan jar file.
   - `<AGENT_VERSION>` is the latest version of the 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>
   </plugin>
   ```
Chapter 2  Installing and Configuring Standard Plugins
Plugins for Package Managers and Build Tools

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>The version of the code-insight-maven-scan-&lt;VERSION&gt;.jar file included with the current plugin (for example, 1.0.2).</td>
</tr>
<tr>
<td>fnciServer</td>
<td>(Required) The URL path to the FlexNet Code Insight server in the following format: http://&lt;CODE_INSIGHT_SERVER_HOST_NAME&gt;:&lt;PORT_NUMBER&gt;/codeinsight/</td>
</tr>
<tr>
<td>fnciAuthToken</td>
<td>(Required) The JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it here. Be sure to include the command “Bearer” followed by the token value, as in the example: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsInVzZXJJZCI6MSwia</td>
</tr>
<tr>
<td>fnciProjectName</td>
<td>(Required) The name of the FlexNet Code Insight inventory-only project created on the Code Insight server for your application codebase scans.</td>
</tr>
<tr>
<td>pluginRootPath</td>
<td>Currently not used.</td>
</tr>
<tr>
<td>pluginProjectName</td>
<td>(Optional) The name of the application project being scanned. This name will appear, along with the Code Insight project name, in the Last Scan field on the on the Summary 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.</td>
</tr>
<tr>
<td>pluginDescription</td>
<td>(Optional) A description of the application project being scanned. This text will appear in the Description field on the Summary tab for the project in the FlexNet Code Insight user interface.</td>
</tr>
<tr>
<td>pluginPathPrefix</td>
<td>(Optional) The path prefix for the codebase files being scanned. This prefix is used to reference the codebase file paths on the Project report generated from the Summary tab for the project in the FlexNet Code Insight user interface.</td>
</tr>
</tbody>
</table>

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.
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.

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-insightscan: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.

<table>
<thead>
<tr>
<th>Task</th>
<th>To install the Artifactory plugin, do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Download and extract the Artifactory plugin subfolder from the CodeInsightversionPlugins.zip file. For more information, see Downloading Plugins.</td>
</tr>
<tr>
<td>2.</td>
<td>Copy the following directory and files into the &lt;ARTIFACTORY_HOME&gt;/etc/plugins directory on the Artifactory server:</td>
</tr>
<tr>
<td></td>
<td>• libs directory</td>
</tr>
<tr>
<td></td>
<td>• code-insight-scan-plugin.groovy</td>
</tr>
<tr>
<td></td>
<td>• code-insight-scan.plugin.props</td>
</tr>
<tr>
<td>3.</td>
<td>Define the properties in the code-insight-scan.plugin.props file:</td>
</tr>
<tr>
<td></td>
<td>repoKeys=&lt;REPOSITORY_PATH1&gt;,&lt;REPOSITORY_PATH2&gt;</td>
</tr>
<tr>
<td></td>
<td>codeinsight.server= http(s)://&lt;HOST&gt;:&lt;PORT&gt;/</td>
</tr>
<tr>
<td></td>
<td>codeinsight.auth.token=Bearer &lt;JWT_TOKEN&gt;</td>
</tr>
<tr>
<td></td>
<td>codeinsight.project.name= &lt;PROJECT_NAME1&gt;,&lt;PROJECT_NAME2&gt;</td>
</tr>
<tr>
<td></td>
<td>plugin.root.path=./artifactory-pro-5.10.2/etc/plugins</td>
</tr>
<tr>
<td></td>
<td>plugin.project.description= will be set by plugin, can be left blank</td>
</tr>
<tr>
<td></td>
<td>isScanCronJobEnabled=disabled</td>
</tr>
<tr>
<td></td>
<td>isPluginEnabled=enabled</td>
</tr>
<tr>
<td></td>
<td>cronJobTime=1 * * * * ?</td>
</tr>
<tr>
<td></td>
<td>artifactory_url= http(s)://&lt;HOST&gt;:&lt;PORT&gt;/ARTIFACTORY/</td>
</tr>
<tr>
<td>4.</td>
<td>Determine if you want to execute the scan with a cron job or by calling REST API:</td>
</tr>
<tr>
<td></td>
<td>• To execute a scan with a cron job, see Scanning an Artifactory Repository Using a Cron Job.</td>
</tr>
<tr>
<td></td>
<td>• To execute a scan by calling REST API, see Scanning an Artifactory Repository Using REST API.</td>
</tr>
</tbody>
</table>

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.

```
0 30 6 * * ? --> Scan trigger that will fire daily at 6:30 am
```

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"
```
Chapter 2  Installing and Configuring Standard Plugins

Plugins for Container Platforms

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:

```
```

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.

- The Docker image you are scanning must already be downloaded to your system.
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 CodeInsight*version*Plugins.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.eyJzdWIiOiJhZG1pbiIsInVzZXJJZCI6MiwiTGF0IGJhc2giOiJmNzIzOWZjOWEtZTM4Yy00MjMwLWFmNGMtYmQ5YTE3YjQ2NjE0IiwiaWF0IjoxNDExMDU3MjI5LCJleHBpcmVfdHMiOjEwMCJ9
   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 JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it here. Be sure to include the command “Bearer” followed by the token value, as in the example:

     ```
     Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsInVzZXJJZCI6MiwiTGF0IGJhc2giOiJmNzIzOWZjOWEtZTM4Yy00MjMwLWFmNGMtYmQ5YTE3YjQ2NjE0IiwiaWF0IjoxNDExMDU3MjI5LCJleHBpcmVfdHMiOjEwMCJ9
     ```

     For more information about generating this token, see Providing an 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.
• 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
• Enabling the Generic Scan Agent Plugin to Detect Transitive Dependencies

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)
Chapter 2  Installing and Configuring Standard Plugins

Generic Scan Agent Plugin

- 64-bit Java Runtime Environment JRE 8 or later
- FlexNet Code Insight Core Server
- 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.

**Note** • 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.

**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. If you want the generic scan agent plugin to detect transitive dependencies during scans, follow the procedure described in Enabling the Generic Scan Agent Plugin to Detect Transitive Dependencies to configure this capability.
4. To execute a scan using the plugin, 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>`—The build version of the .jar file used to run the scan agent. The version is shown in the name of the .jar file, which is located in the code-insight-agent-sdk-generic/generic-plugin-binary folder.
   - `<FNCI_SERVER_HOSTNAME>:<PORT>/<FNCI_SERVER_PATH>`—The URL for the FlexNet Code Insight Core Server (for example, http://1.1.1.1:8888/codeinsight)
   - `<JWT_TOKEN>`—Your JSON Web Token (JWT) used to authorize user access to the Code Insight functionality. Generate this token using the Code Insight Web UI and then copy and paste it in this field. For more information, see Providing an Authorization Token.
Enabling the Generic Scan Agent Plugin to Detect Transitive Dependencies

Use this procedure to enable the detection of transitive dependencies during scans performed by the generic scan agent plugin. If you do not create and deploy the file described in this procedure, no transitive-dependency detection is performed.

To enable the generic scan agent plugin to detect transitive dependencies, do the following:

1. Create a text file whose content contains the following properties. These properties enable the various Code Insight analyzers to detect transitive dependencies during a scan:

   - `flx.enable.gradle.file.transitive.dependency=true`
   - `flx.enable.npm.transitive.dependency=true`
   - `flx.enable.nuget.transitive.dependency=true`
   - `flx.enable.java.transitive.dependency=true`

   If you do not want a specific analyzer to detect transitive dependencies, set the boolean value for its corresponding property to `false` (or remove the property from the file content). See the table below for more information.

2. Save the file as `codeaware.properties`.
3. Copy the file to the `code-insight-agent-sdk-generic/generic-plugin-binary` folder (where the `codeinsight-generic-build.jar` file is located).

Table 2-3 • Properties Enabling the Generic Scan Agent Plugin to Detect Transitive Dependencies During Scans

<table>
<thead>
<tr>
<th>Package Analyzer</th>
<th>Property (with Boolean value) to Enable Transitive Dependency Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradle</td>
<td>`flx.enable.gradle.file.transitive.dependency={true</td>
</tr>
<tr>
<td>NPM</td>
<td>`flx.enable.npm.transitive.dependency={true</td>
</tr>
<tr>
<td>NuGet</td>
<td>`flx.enable.nuget.transitive.dependency={true</td>
</tr>
<tr>
<td>Java (pom.xml)</td>
<td>`flx.enable.java.transitive.dependency={true</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Settings/Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecutionContext</td>
<td>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 ExecutionContext.getInstance(...) method.</td>
<td>Methods:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ExecutionContext getInstance(Properties props, PrintStream logger)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>props</strong>: Properties required to scan and publish results</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>logger</strong>: 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 Property Settings.</td>
</tr>
</tbody>
</table>
Table 3-1 • Available Classes and Methods

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Settings/Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ScanExecutor</strong></td>
<td>Executes the scan and the publish results request made by client plugins.</td>
<td><strong>Methods:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ScanExecutor(ExecutionContext executionContext)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A constructor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>String testConnection()</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Validates the user authorization token and the connection to the FlexNet Code Insight server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Valid returned strings:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Server Not Found</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Invalid Auth Token Found</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• User not authorized to access the project</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>String scanCodebase(List&lt;String&gt; paths)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 ExecutionContext.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>paths: list of absolute paths to be scanned</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Return:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUCCESS or FAILURE as a string. See the logger output stream for more details.</td>
</tr>
</tbody>
</table>

### Property Settings

The following table lists property settings that you can update:

Table 3-2 • Property Settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Required/Optional?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>codeinsight.server</td>
<td>Required</td>
<td>Code Insight server URI (for example, http(s)://your.codeinsight.server:port/).</td>
</tr>
<tr>
<td>codeinsight.auth.token</td>
<td>Required</td>
<td>Authorized user token generated in the FlexNet Code Insight user interface.</td>
</tr>
<tr>
<td>codeinsight.project.name</td>
<td>Required</td>
<td>The FlexNet Code Insight project name where all the inventory information will be published.</td>
</tr>
<tr>
<td>plugin.root.path</td>
<td>Required</td>
<td>Local root path that can be replaced with path prefix.</td>
</tr>
</tbody>
</table>
Chapter 3  Developing Custom Plugins

Downloading the Scan Agent Toolkit

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

<table>
<thead>
<tr>
<th>Task</th>
<th>To download the Scan Agent toolkit, do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Log into the Customer Community page of the Flexera website: <a href="https://flexeracommunity.force.com/customer/">https://flexeracommunity.force.com/customer/</a></td>
</tr>
<tr>
<td>2.</td>
<td>Click Downloads.</td>
</tr>
<tr>
<td>3.</td>
<td>Click the Access button under FlexNet Code Insight. The Product and License Center page appears.</td>
</tr>
<tr>
<td>4.</td>
<td>Select FlexNet Code Insight from the Your Downloads list.</td>
</tr>
<tr>
<td>5.</td>
<td>Select the version of FlexNet Code Insight from the list. The Downloads page appears.</td>
</tr>
<tr>
<td>7.</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

---

### Table 3-2 • Property Settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Required/Optional?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plugin.project.name</td>
<td>Optional</td>
<td>Name of the plugin (for example, “Codeinsight Jenkins plugin”).</td>
</tr>
<tr>
<td>plugin.project.description</td>
<td>Optional</td>
<td>Plugin’s instance name/build number or tag can be sent to the Code Insight server.</td>
</tr>
<tr>
<td>plugin.path.prefix</td>
<td>Optional</td>
<td>Prefix to be added to file paths for display to the user. For example, it could be the URL for a Jenkins workspace.</td>
</tr>
</tbody>
</table>
Writing a Custom Scan Agent Plugin

Writing a custom scan agent plugin involves the following tasks:

- **Task 1**—Review the example generic scan agent plugin for its code structure and build configuration. (In essence, you can use this plugin as a template for creating your own.)
- **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 codebase 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.
Chapter 3  Developing Custom Plugins
Deploying a Custom Scan Agent Plugin