OpenText™ ScanCentral SAST

Installing, Configuring, and Using OpenText™ ScanCentral SAST

Version : 25.4

PDF Generated on : 28/10/2025

Table of Contents

1. Installing, Configuring, and Using OpenText™ ScanCentral SAST

This guide provides information on how to install, configure, and use OpenText ScanCentral SAST to streamline your static code analysis process.

1.1. Change log

The following table lists changes made to this document. Revisions to this document are published only if the changes made affect product functionality.

Software release / Document version	**Changes**
25.4.0	Added: • Ability to configure client and sensor logging options using environment variables (see Configuring the client logging and Configuring the sensor logging) • Deploying the Controller on Tomcat • Ability to configure the installation location of OpenText SAST using environment variable (see About OpenText ScanCentral SAST sensors and Installing a sensor as a service Installing a sensor as a service) • Added definition of embedded client (see Embedded clients and standalone clients) • Option to customize the configuration for Debricked CLI (see Package command and Open source software composition analysis (OpenText Core Application Security only)) Updated: • Changed the procedure to enable OpenText ScanCentral SAST sensor auto-start on a Linux system (see Enabling sensor auto-start on a Linux system) • Instructions of Installing a sensor using OpenText SAST Removed: • The "Installing an embedded client" topic was removed
25.2.0/Revision1: July 25,2025	Added: Options `-sastver`, `--sast-version` for the `start` command to assign the remote translation and scan jobs to a specific version of OpenText SAST (see Start command)

25.2.0	Added: • Ability to configure Controller logging options using environment variables (see Configuring the Controller logging) • Ability to prevent the restoration of dependencies during packaging for scan requests by including the `-skipBuild` option (available for Go, JavaScript/TypeScript, PHP, and Python projects) Updated: • Changed the location of where to download the Helm charts (see Optional Kubernetes and Docker deployment) • Added a property to specify if job status is included in the email notification subject (see Configuring the Controller and Requesting job status email notifications for scan requests) • By default, the Controller is configured to replace duplicate scan jobs (see Configuring the Controller) • You can specify the files to include for analysis in the `start` and `package` commands (see Submitting remote translation and scan requests, Start command, and Package command) • Changed the default OpenText Core SCA CLI location (see Open source software composition analysis ([%=FortifyProducts_Vars.FoD%] only)) • You can perform only translation (no scan) on a project package with PackageScanner (see Using the PackageScanner tool) • You can specify multiple email addresses for job status email notifications (see Requesting job status email notifications for scan requests and Start command) • The `--include-test` option applies to .NET projects (test projects for .NET code are excluded by default) (see Start command, and Package command) • You can use a pool name to specify a sensor pool with the `start` and `worker` commands (see Targeting a specific sensor pool for a scan request, Start command, and Worker command) • Added supported Python options (see Options accepted for -targs (--translation-args))

24.4.0	Added: • Configuring licensing with Fortify License and Infrastructure Manager • Scanning JavaScript and TypeScript code • Instructions for scanning PHP projects that use Composer (see Scanning PHP projects) • You can add JVM system and OpenText ScanCentral SAST properties (for clients and sensors) to the client commands by adding the `-D` option to an environment variable (see Configuring client properties, and Configuring sensor properties) Updated: • You can retrieve job files using the retrieve command (see Retrieve command) • You can add JVM system properties to an environment variable for use with PackageScanner (see Using the PackageScanner tool) • The `start` command `-uptoken` option is no longer required to upload scan results to Application Security if you include the global `-ssctoken` (see Uploading results to Application Security) • Described the OpenText ScanCentral SAST Controller service account for uploading scan results to Application Security (see Uploading results to Application Security) • Added supported `-gotags` option (see Options accepted for -targs (--translation-args)) • Added supported `-bin` option (see Options Accepted for -sargs (--scan-args)) Removed: • The `-hello` option for the `worker` command is ignored and was removed from this help. This option will be removed from the product in a future release. • The `--scan-node-modules` option for the `start` and `package` commands is ignored and was removed from this help. This option will be removed from the product in a future release.
24.2.0	Added: • Option to replace duplicate scan requests that are uploaded to the same application version in Application Security (see Configuring the Controller, and Preventing Replacement of Duplicate Scan Requests) • Option to configure the Controller to assign scan jobs to a specific version of OpenText SAST (see Configuring the Controller) • (For use with OpenText Core Application Security only) Ability to use the Debricked CLI for open source software composition analysis (see Generating a OpenText ScanCentral SAST package, Open Source Software Composition Analysis (Fortify on Demand Only), and Package Command) Updated: • MSBuild and dotnet build logs are included in the debug archive (see Creating a Log Archive for Customer Support) • The options to display the version are `-v` and `--version`. The `-version` option is deprecated (see Global Options) • The `--php-version` option for the `start` and `package` commands is no longer required because OpenText ScanCentral SAST automatically detects the installed PHP version (see Start Command Options and Package Command Options) • The option `--output` for the `package` command is no longer required (see Package Command Options) Removed: • The "Working with Salesforce Apex Projects" topic was removed because the `-apex` OpenText SAST option is no longer required to analyze Apex projects.

23.2.0	Added: • Optional Kubernetes and Docker Deployment • Installing the Controller as a Service on Linux • Managing the Controller Service on Linux • OpenText ScanCentral SAST API • Working with COBOL Projects and added supported COBOL-related options (see Options Accepted for -targs (--translation-args)) • Retrying Failed Uploads to Application Security • Preserving the OpenText SAST Project Root Directory • Creating Archive Logs for Customer Support Updated: • Changed the requirements for when to run the migration script to upgrade the OpenText ScanCentral SAST Controller (see Upgrading the Controller) • Updates for analyzing .NET projects (see Configuring Sensors to Offload Translation for .NET Languages and Working with .NET Projects) • Added descriptions of the scan status values (see Viewing a Scan Request Status) • Added supported `-scan-policy` option (see Options Accepted for -sargs (--scan-args)) • Added supported COBOL options (see Options Accepted for -targs (--translation-args)) Removed: • The `arguments` command is deprecated and removed from this helpdocument. Use the `-targs` or `-sargs` option with the `start` or `package` commands instead.

1.2. Introduction

With OpenText ScanCentral SAST, users of OpenText™ Static Application Security Testing (OpenText SAST) can better manage their resources by offloading code analysis tasks from their build machines to a distributed network of computers (sensors) configured for this purpose. In addition to freeing up build machines, this process enables you to add more resources to the scan machines as needed, without having to interrupt the build process. The command-line interface enables integration of static analysis with the build process and provides the ability to dynamically scale the sensors needed to perform the work required of the CI/CD pipelines with respect to running scans.

There are two ways to start an OpenText SAST analysis of your code from a OpenText ScanCentral SAST client:

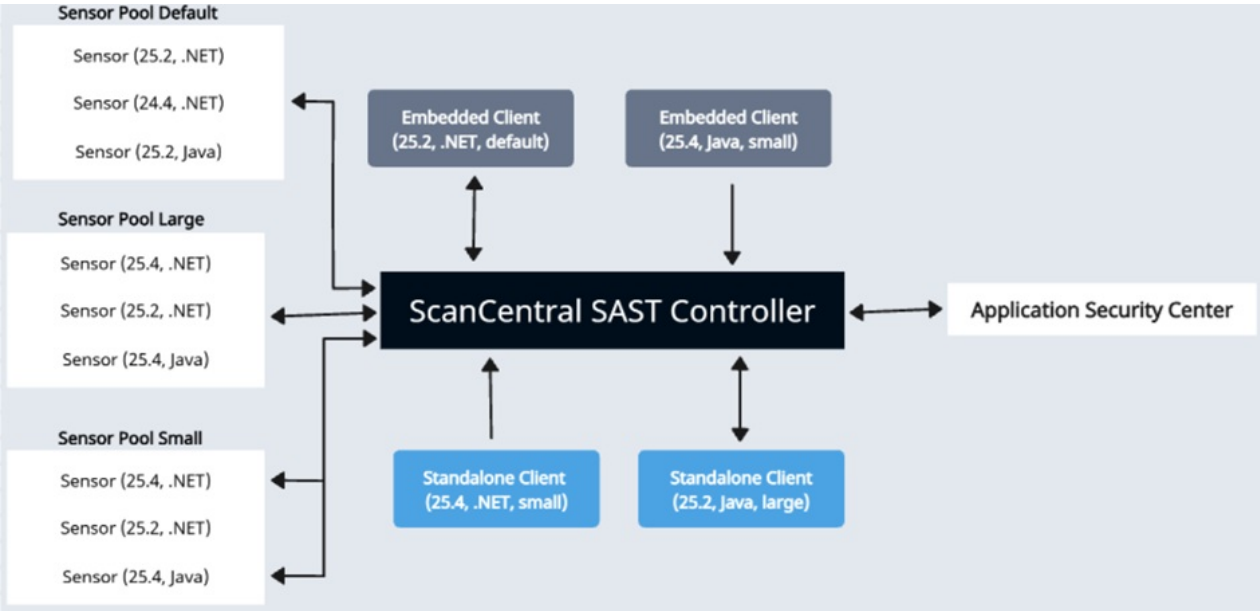
- Remote Translation and Scan—Offload the entire analysis to the sensors. Your application must be written in a language supported for remote translation. For a list of supported languages, see the *Application Security Software System Requirements* document. If your code is written in a language other than those supported in remote translation, then you must perform a local translation and remote scan.
- Local Translation and Remote Scan—Perform the translation phase (less processor- and time-intensive than the scan phase) on a local or build machine. After the translation is complete, use OpenText ScanCentral SAST client to move the OpenText SAST mobile build session (MBS) to sensors to scan.

This content provides information on how to install, configure, and use OpenText ScanCentral SAST to streamline your static code analysis process.

This section contains the following topics:

1.2.1. OpenText ScanCentral SAST components

The following diagram illustrates a OpenText™ ScanCentral SAST environment.



A OpenText™ ScanCentral SAST deployment includes the following three components:

Note

The minimum deployment requires three physical or virtual machines: a Controller, a sensor, and a client. An OpenText™ Application Security server is optional.

- **OpenText ScanCentral SAST Controller**—A standalone web application that receives project packages with translation and scan instructions (or OpenText SAST mobile build sessions (MBS) and scan instructions from OpenText ScanCentral SAST clients), routes the information to sensors, and (optionally) uploads scan results (FPR files) to Application Security. For more detail, see About the OpenText ScanCentral SAST Controller.
- **OpenText ScanCentral SAST sensors**—A distributed network of computers set up to receive scan requests and analyze code using OpenText SAST. A sensor accepts either a mobile build session (MBS) file and performs a scan, or it accepts a project package that contains sources and dependencies, which it translates and scans. For more information, see About OpenText ScanCentral SAST sensors.

  To scan code, sensors must belong to a **sensor pool**. A sensor pool consists of one or more sensors, grouped based on any criteria, which you can then target for scan requests. For example, you can create a sensor pool that consists of machines with a lot of physical memory to use for scan requests that require a lot of memory. If you do not specifically add a sensor to a sensor pool, it is automatically assigned to the default sensor pool.

- **OpenText ScanCentral SAST client**— On a build machine, clients can generate packages for remote translation and scan independent of OpenText SAST. Clients can also be run on a build machine on which OpenText SAST translates code and generates mobile build sessions (MBS). The translated source code, along with optional and required data, such as custom rules and OpenText SAST command-line options, are uploaded to the Controller for analysis. For more information, see About OpenText ScanCentral SAST clients.

To successfully deploy OpenText ScanCentral SAST, complete the following tasks in the order listed:

- (Recommended, but not required) Deploy a (or connect to an existing) Application Security instance

  For more information, see Working with Application Security.

- Install the OpenText ScanCentral SAST Controller
- Install OpenText ScanCentral SAST sensors
- Install OpenText ScanCentral SAST clients

The following sections provide instructions for completing these tasks. For information about hardware and software requirements for these components, see the *Application Security Software System Requirements* document.

1.2.2. Working with Application Security

Although you can deploy a standalone OpenText ScanCentral SAST Controller, communication with Application Security provides the following additional benefits:

- The Controller can upload scan results directly to Application Security application versions.
- The Application Security user interface includes a **ScanCentral** view where you can:
    - View Controller and sensor information
    - View scan request details and export scan results and log files
    - Create and manage sensor pools to which you can target scan requests
    - Prioritize and cancel scan requests
    - Place the Controller in maintenance mode (see Placing the Controller in Maintenance Mode)
    - Shut down sensors (see Safely shutting down sensors)

For instructions on how to integrate OpenText ScanCentral SAST with Application Security, see the *OpenText™ Application Security User Guide*.

Note

If you have Application Security with <year>.<quarter> version, you can deploy OpenText ScanCentral SAST with either the same version or one version higher.

For example, if you have 24.4 Application Security, you can deploy either 24.4 or 25.2 OpenText ScanCentral SAST.

1.2.3. Securing OpenText ScanCentral SAST deployment

The OpenText Application Security Software products collect and display information about an enterprise’s applications. That information includes summaries of the potential security vulnerabilities uncovered in the source code.

Just as you apply security precautions to your applications, you must also secure access to the OpenText ScanCentral SAST components. The security vulnerability summaries that OpenText products provide might mandate an even higher level of secure deployment.

OpenText ScanCentral SAST works with your codebase. Because this information allows for some opportunities of mishandling or abuse, OpenText recommends that you deploy OpenText ScanCentral SAST in a secure operations facility and secure access to the OpenText ScanCentral SAST installation directories.

1.2.4. Securing Tomcat server

You must ensure the operational security of Apache® Tomcat™ server. At a minimum, configure Tomcat server to use HTTPS in conjunction with an SSL certificate issued by a trusted certificate authority. Also, take any additional steps necessary to secure Tomcat server in your operating environment.

OpenText recommends that you use secure SSL/TLS cipher suites in Tomcat.

- APR-based SSL connections

  Use the SSLCipherSuite directive. For detailed information, go to the SSL CipherSuite Directive and Cipher Suites and Enforcing Strong Security webpages.

- JSSE-based SSL connections

  Use the `ciphers` and the `honorCipherOrder` attributes. For details, see the Apache Tomcat 10 Configuration Reference.

Because of trade-offs between improved security and improved interoperability, better performance, and so on, there is no correct cipher suite choice. However, Apache provides information that can help you make that choice in the Apache Tomcat Ciphers documentation.

1.2.5. Optional Kubernetes and Docker deployment

This content describes how to install OpenText ScanCentral SAST without using a Kubernetes cluster or Docker®. To use Kubernetes for OpenText ScanCentral SAST container orchestration, Helm charts are available on Docker Hub at https://hub.docker.com/r/fortifydocker/helm-scancentral-sast.

Note

Helm charts might not be available immediately after product release. When Helm charts for the current release are available, Helm chart documentation will be available on the Application Security Documentation website.

OpenText provides OpenText ScanCentral SAST images that you can download from Docker Hub. Access to the Fortify Docker repository requires credentials and is granted through your Docker ID. To access the Fortify Docker repository, email your Docker ID to mfi-fortifydocker@opentext.com.

1.2.6. Related documents

This topic describes documents that provide information about OpenText Application Security Software products.

Note

Most guides are available in both PDF and HTML formats. Product help is available within the Fortify License and Infrastructure Manager (LIM) and the OpenText DAST products.

All products

The following documents provide general information for all products. Unless otherwise noted, these documents are available on the Product Documentation website for each product.

Document / file name	Description
About OpenText Application Security Software Documentation appsec-docs-n-*<version>*.pdf	This paper provides information about how to access OpenText Application Security Software product documentation. This document is included only with the product download.
OpenText™ Application Security Software System Requirements appsec-sr-*<version>*.pdf	This document provides the details about the environments and products supported for this version of OpenText Application Security Software.
What's New in OpenText Application Security Software <version> appsec-wn-*<version>*.pdf	This document describes the new features in OpenText Application Security Software products.
OpenText Application Security Software Release Notes appsec-rn-*<version>*.pdf	This document provides an overview of the changes made to OpenText Application Security Software for this release and important information not included elsewhere in the product documentation.

Application Security

The following document provides information about OpenText Application Security (Software Security Center). This document is available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-software-security-center.

Document / file name	Description

Document / file name	Description
OpenText™ Application Security User Guide ssc-ugd-*<version>*.pdf	This document provides Application Security users with detailed information about how to deploy and use Application Security. It provides all the information you need to deploy, configure, and use Application Security. It is intended for use by system and instance administrators, database administrators (DBAs), enterprise security leads, development team managers, and developers. Application Security provides security team leads with a high-level overview of the history and status of a project.

OpenText SAST

The following documents provide information about OpenText SAST (Fortify Static Code Analyzer). Unless otherwise noted, these documents are available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-static-code.

Document / file name	Description
OpenText™ Static Application Security Testing User Guide sast-ugd-*<version>*.pdf	This document describes how to install and use OpenText SAST to scan code on many of the major programming platforms. It is intended for people responsible for security audits and secure coding.
OpenText™ Static Application Security Testing Custom Rules Guide sast-cr-ugd-*<version>*.zip	This document provides the information that you need to create custom rules for OpenText SAST. This guide includes examples that apply rule-writing concepts to real-world security issues. **Note** This document is included only with the product download.
OpenText™ Fortify License and Infrastructure Manager Installation and Usage Guide lim-ugd-*<version>*.pdf	This document describes how to install, configure, and use the Fortify License and Infrastructure Manager (LIM), which is available for installation on a local Windows server and as a container image on the Docker platform.

1.3. About the OpenText ScanCentral SAST Controller

The OpenText ScanCentral SAST Controller (Controller) is a standalone server that sits between the OpenText ScanCentral SAST clients, sensors, and optionally, Application Security. The Controller accepts scan requests issued by clients and assigns them to an available sensor. A sensor returns scan results to the Controller, which stores them temporarily.

This section contains the following topics:

1.3.1. Installing the Controller

For information about how to upgrade your Controller, see Upgrading OpenText ScanCentral SAST Components and Upgrading the Controller.

Important

- Before you install the Controller, you must first download and configure a supported Java™ Runtime Environment (JRE). For information about supported JRE versions, see Controller application server. For information about how to download and configure a JRE, see the documentation for the supported JRE version.
- To install the Controller as a Microsoft Windows® or Linux® service, make sure that you extract the contents in a directory where the local service (Windows) or the user or group using the service (Linux) has access.
- The name of the directory into which you install the Controller must not include spaces.

1.3.1.1. Installing the Controller with included Tomcat

To install the Controller (on a Windows or Linux system):

- Extract the contents of the `Fortify_ScanCentral_Controller_<version>_x64.zip` file into a directory of your choosing.

In this guide, `<controller_install_dir>` refers to the Controller installation directory and `<sast_install_dir>` refers to the OpenText SAST installation directory.

After you install the Controller, the `<controller_install_dir>` resembles the following:

```
bin/
db-migrate/
tomcat/
readme.txt
```

1.3.1.2. Deploying the Controller on Tomcat

To deploy the Controller on your Tomcat instance:

1. Extract the contents of the installation package into a temporary directory in a secure location.
2. Locate the distribution file (`Fortify_ScanCentral_Controller_WAR_Tomcat_<version>_x64.zip`) and extract all the contents into a directory in a secure location.

Note

The directory into which you extract the distribution file content is referred to in all topics as the `<controller_distribution_dir>` directory.

3. Copy the `<controller_distribution_dir>/client/` directory (and contents) to the `<tomcat>/` directory.
4. Copy the `scancentral-ctrl.war` file to the `<tomcat>/webapps/` directory.
5. Start the Tomcat server.

1.3.1.3. Installing the Controller as a Windows service

To install the Controller as a service on a Windows machine without other Tomcat instances running:

1. Log on to Windows as a local user with administrator permissions.
2. Make sure that the `JRE_HOME` and `JAVA_HOME` environment variables are correctly configured.
3. Make sure that the `CATALINA_HOME` environment variable is either empty or set up to point to the `<controller_install_dir>\tomcat` directory.
4. Go to `<controller_install_dir>\tomcat\bin`, and then run the following command:

```
service.bat install
```

This creates a service with the name `Tomcat10`.

To install the Controller as a service with a different name:

1. Make sure that the `JRE_HOME` and `JAVA_HOME` environment variables are correctly configured.
2. Make sure that the `CATALINA_HOME` environment variable is either empty or set up to point to the `<controller_install_dir>\tomcat` directory.
3. Go to `<controller_install_dir>\tomcat\bin\`, and then run the following command:

```
service.bat install <service_name>
```

Important

The service name must not contain any spaces.

Configuring Java memory for the service

To configure the Java memory for the Controller service:

1. Run `tomcat10w.exe`.
2. In the Apache Tomcat Properties window, click the **Java** tab, and then set the **Maximum memory pool** value.
3. Restart the service.

1.3.1.4. Uninstalling the Controller Windows service

To uninstall the Apache Tomcat 10 service for the Controller:

1. Stop the service.
2. Go to `<controller_install_dir>\tomcat\bin\`, and then run the following command:

```
service.bat remove
```

To uninstall the Controller as a service with a name other than `Tomcat10`:

1. Stop the service.
2. Go to `<controller_install_dir>\tomcat\bin\`, and then run the following command:

```
service.bat remove <service_name>
```

1.3.1.5. Installing the Controller as a service on Linux

You can install the OpenText ScanCentral SASTController as a service on Linux. These instructions provide an example of one method of installing the Controller as a service.

To install the Controller as a service on a Linux system:

1. Install the Controller in a location where the user and group using the service have access.

   For installation instructions, see Installing the Controller.

2. Configure the Controller service by creating a systemd unit file `scancentral.service` in the `/etc/systemd/system/` directory with the following content.

   In the following content, replace *<controller_install_dir>* with the directory where you installed the Controller in step 1. Replace *<path_to_jre>* with the location of your JRE.

   ```
   [Unit]
   Description=ScanCentral SAST Controller Service
   After=syslog.target network.target

   [Service]
   Type=forking
   #User to run ScanCentral SAST Controller. If commented out, the root user is used.
   #User=sc_user
   #Group to run ScanCentral SAST Controller. If commented out, the root group is used.
   #Group=sc_user

   #Specify the location of JRE
   Environment=JAVA_HOME=<path_to_jre>
   Environment=CATALINA_PID=<controller_install_dir>/tomcat/temp/tomcat.pid
   Environment=CATALINA_HOME=<controller_install_dir>/tomcat
   Environment=CATALINA_BASE=<controller_install_dir>/tomcat
   #Uncomment and specify CATALINA_OPTS if needed
   #Environment=CATALINA_OPTS=
   #Uncomment and specify JAVA_OPTS if needed
   #Environment=JAVA_OPTS=

   ExecStart=<controller_install_dir>/tomcat/bin/startup.sh
   ExecStop=/bin/kill -15 $MAINPID

   [Install]
   WantedBy=multi-user.target
   ```

3. Reload the daemon to discover and load the new service file:

   ```
   systemctl daemon-reload
   ```

4. Enable the service to start on startup by running the following command:

   ```
   systemctl enable scancentral.service
   ```

See also

Administering the Controller Service on Linux

1.3.1.6. Managing the Controller service on Linux

To manage the Controller service, run the following command:

```
service scancentral [start | stop | restart | status]
```

or you can use Systemd directly:

```
systemctl [start | stop | restart | status] scancentral
```

See also

Installing the Controller as a Service on Linux

1.3.2. Specifying the Controller web address

In this help, *<controller_url>* refers to a correctly formatted OpenText ScanCentral SAST Controller web address. The correct format for the Controller web address is as follows:

<protocol>://*<controller_host>*:*<port>*/scancentral-ctrl

1.3.3. Securing the Controller

This topic describes how to create a secure connection (HTTPS) between the OpenText ScanCentral SASTController/Tomcat server and the OpenText ScanCentral SAST client. This procedure requires either a self-signed certificate or a certificate signed by a certificate authority such as VeriSign.

Note

These instructions describe a third-party product and might not match the specific, supported version you are using. See your product documentation for the instructions for your version.

Creating a secure connection using self-signed certificates

To enable SSL on Tomcat using a self-signed certificate:

1. To generate a keystore that contains a self-signed certificate, open a command prompt and run the following Java keytool command:

   ```
   keytool -genkey -alias <alias_name> -keyalg RSA -keystore <mykeystore>
   ```

2. Provide values for the prompts as described in the following table.

Prompt	Description
Enter keystore password:	Type a secure password.
Re-enter new password:	Re-type your secure password.
What is your first and last name?	Type your hostname. You can use your fully qualified domain name here. **Note** To provide an IP address as the hostname, you must also provide the `-ext san=ip:<ip_address>` option to keytool. Without this additional option, the SSL handshake fails.
What is the name of your organizational unit?	Name to identify the group that is to use the certificate.
What is the name of your organization?	Name of your organization.
What is the name of your City or Locality?	City or locality in which your organization is located.
What is the name of your State or Province?	State or province in which your organization is located.
What is the two-letter country code for this unit?	For example, if your server is in the United States, type `US`.
Confirm your entries:	Type `yes` to confirm your entries.

Enter key password for <tomcat><Return if same as keystore password>:	Password for your Tomcat server key or press **Enter** to use the same password you established for your keystore. OpenText recommends that you create a new key password.
Re-enter new password:	Re-type your key password.

3. To export the certificate from the Tomcat keystore, open a command prompt and type the following:

```
keytool -export -alias <alias_name> -keystore <mykeystore> -file "YourCertFile.cer"
```

4. Add the following connector to the `server.xml` file in the `tomcat/conf` directory:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
 maxThreads="150" SSLEnabled="true" maxParameterCount="1000">
 <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
 <SSLHostConfig>
  <Certificate certificateKeystoreFile="<my_keystore>"
   certificateKeystorePassword="<my_password>" type="RSA" />
 </SSLHostConfig>
</Connector>
```

   The default `server.xml` file installed with Tomcat includes an example `<Connector>` element for an SSL connector.

5. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor:

6. Update the `this_url` property with your HTTPS address and port as shown in the following example:

```
this_url=https://<controller_host>:8443/scancentral-ctrl
```

7. Restart your Tomcat server.
8. Set up your clients and sensors.

   For information about how to set up the OpenText ScanCentral SAST clients and sensors, see Installing clients and Installing sensors, respectively.

9. Add your self-signed certificate to the Java keystore on all entities that communicate with the Controller (includes all clients, sensors, and Application Security installations) as follows:

   1. For embedded clients and sensors, go to the `<sast_install_dir>/jre/bin/` directory where *<sast_install_dir>* is the directory where the sensor or client is installed.
   2. For the installation of standalone clients, type one of the following commands:
      - On a Windows system: `cd %JAVA_HOME%\jre\bin`
      - On a Linux system: `cd $JAVA_HOME/jre/bin`
   3. Run the following command:

      ```
      keytool -importcert -alias <aliasName> -keystore ../lib/security/cacerts -file "YourCertFile.cer" –trust cacerts
      ```

      where `YourCertFile.cer` is the same certificate file that you exported in step 3.

Creating a secure connection using a certificate signed by a certificate signing authority

To enable SSL on Tomcat using a certificate signed by a certificate signing authority:

1. Use the Java keytool to generate a new keystore containing a self-signed certificate:

   ```
   keytool -genkey -alias <alias_name> -keyalg RSA -keystore <mykeystore>
   ```

2. The keytool prompts you for the information described in the following table.

Prompt	Description
Enter keystore password:	Type a secure password.
Re-enter new password:	Re-enter your secure password.
What is your first and last name?	Type your hostname. You can use your fully qualified domain name here. **Note** To enter an IP address as the hostname, you must also pass an additional option to keytool, `-ext san=ip:<ip_address>`. Without this additional option, the SSL handshake fails.
What is the name of your organizational unit?	Type the name of the group that is to use the certificate.
What is the name of your organization?	Type the name of your organization.
What is the name of your City or Locality?	Type the city or locality.
What is the name of your State or Province?	Type the state or province.
What is the two-letter country code for this unit?	If your server is in the United States, type `US`.
Confirm your entries:	Type `yes` to confirm your entries.
Enter key password for `<tomcat><Return if same as keystore password>`:	Type a password for your Tomcat server key, or press **Return** to use the same password you established for your keystore. OpenText recommends that you create a new password.
Re-enter new password:	Re-type your key password.

3. Generate a Certificate Signing Request (CSR).

   To obtain a certificate from a certificate signing authority, you must generate a Certificate Signing Request (CSR). The certificate authority uses the CSR to create the certificate. Create the CSR as follows:

   ```
   keytool -certreq -alias <alias_name> -keyalg RSA -file "yourCSRname.csr" -keystore "<mykeystore>"
   ```

4. Send the CSR file to the certificate signing authority you have chosen.
5. After you receive your certificate from the certificate signing authority, import it into the keystore that you

created, as follows:

```
keytool -importcert -alias <alias_name> -trustcacerts -file "YourVerisignCert.crt" -keystore "<mykeystore>"
```

The root CA already exists in the `cacerts` file of your Java™ Development Kit (JDK), so you are just installing the intermediate CA for your certificate signing authority.

Note

If you purchased your certificate from VeriSign, you must first import the chain certificate. You can find the specific chain certificate on the VeriSign website or click the link for the chain certificate in the email you received from VeriSign with your certificate.

```
keytool -importcert -alias IntermediateCA -trustcacerts -file "chainCert.crt" -keystore "<mykeystore>"
```

6. Add the following Connector element to the `server.xml` file in the `tomcat/config` directory:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
 maxThreads="150" SSLEnabled="true" maxParameterCount="1000">
 <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
 <SSLHostConfig>
  <Certificate certificateKeystoreFile="<my_keystore>"
   certificateKeystorePassword="<my_password>" type="RSA" />
 </SSLHostConfig>
</Connector>
```

The default `server.xml` file installed with Tomcat includes an example `<Connector>` element for an SSL connector.

7. Restart Tomcat server.
8. Open `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
9. Update the `this_url` property with your HTTPS address and port as shown in the following example:

```
this_url=https://<controller_host>:8443/scancentral-ctrl
```

1.3.4. Configuring the Controller

After you install the OpenText ScanCentral SASTController, edit global properties such as the email address to use, the shared secret for the Controller (password that Application Security uses when it requests data from the Controller), the shared secret for clients, and the Application Security web address.

Caution

To avoid potential conflicts, OpenText recommends that you run the Controller on a Tomcat server instance other than the instance that Application Security uses.

To configure the Controller:

1. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
2. Configure the properties described in the following table.

Controller property	Description
`client_auth_token`	Specifies a client authentication token string that contains no spaces or backslashes to secure the Controller for use by authorized clients only. If you prefer not to use plain text, you can use an encrypted shared secret as the value for this property. For instructions on how to encrypt a shared secret, see Encrypting the Shared Secret on the Controller.
`client_auto_update`	If set to `true`, the Controller automatically updates all outdated sensors and clients. For details, see Enabling Automatic Updates of Clients and Sensors.
`client_zip_location`	Specifies the location of the directory that contains OpenText ScanCentral SAST client ZIP files. To enable remote upgrades of one or more client versions, place them in this directory. The default value is `client_zip_location=${catalina.base}/client`.
`db_dir`	Specifies the OpenText ScanCentral SAST database home directory. The default value is `${catalina.base}/cloudCtrlDb`.
`job_file_dir`	Specifies the job storage directory. The default value is: `${catalina.base}/jobFiles`.
`fail_job_if_ssc_upload_data_invalid`	If set to `true`, then before the Controller creates a scan job and assigns it to a sensor, it verifies that the following requirements are true: ◦ The token has not expired If the token expires before the Controller assigns the scan job to a sensor, the scan does not run, and the job fails. ◦ The application version exists in Application Security and is active The default value for this property is `true`.

`job_expiry_delay`	Specifies the number of hours after a job finishes that the job becomes a candidate for cleanup. Cleanup removes the job directory, removes jobs from the database, and removes information about expired sensors from the database so that they are no longer displayed in Application Security. By default, the jobs are deleted from the Controller after 168 hours (or 7 days).
`worker_expiry_delay`	Specifies the amount of time (in hours) after a sensor stops communicating that it becomes a candidate for cleanup. The default is 168 hours (or 7 days).
`cleanup_period`	Specifies the frequency (in minutes) that expired jobs and sensors are cleaned up. The default is `60`.
`lim_server_url`	Specifies the web address for the OpenText™ Fortify License and Infrastructure Manager (LIM) server website. The web address format is `https://<location>:<port>`, where *<location>* is IP address, hostname, or domain name. **Note** If the LIM does not use SSL certificates, the protocol is `http`. For more information about using the LIM for sensors, see Configuring Licensing with Fortify License and Infrastructure Manager.
`lim_license_pool`	Specifies the name of the LIM license pool.
`lim_license_pool_password`	Specifies the password for the LIM license pool. You can either use a plain text password or use the `pwtool_keys_file` property to encrypt this password. For information about how to encrypt your passwords, see Encrypting the Shared Secret on the Controller.
`lim_proxy_url`	Specifies the proxy server to access the LIM server if the sensor is behind a proxy.
`lim_proxy_user`	Specifies the LIM proxy user name if authentication is required for the LIM proxy server. For information about how to encrypt user names and passwords, see Encrypting the Shared Secret on the Controller.
`lim_proxy_password`	Specifies the password for the LIM proxy user. You can either use a plain text password or use the `pwtool_keys_file` property to encrypt this password. For information about how to encrypt your passwords, see Encrypting the Shared Secret on the Controller.
`max_upload_size`	Specifies the maximum size (in megabytes) for files uploaded to the Controller from clients or sensors (for example, log files, result files, and job files).
`pool_mapping_mode`	Configures the mode for mapping scan requests to sensor pools. For information about the valid values for `pool_mapping_mode`, see About the pool_mapping_mode Property.

`pwtool_keys_file`	Specifies the path to a file with pwtool keys. If encrypted passwords are used, this must specify a file with the pwtool keys used to encrypt the passwords.
`scan_timeout`	Specifies the maximum amount of time (in minutes) that sensors can process a scan job and be prevented from doing other jobs. After the specified time has passed, a scan job is canceled. This setting applies to all sensors associated with the Controller but can be overridden with the `--scan-timeout` command-line option for a specific job or sensor (see Setting the Maximum Run Time for Scans and Start Command Options).
`accept_job_when_no_sensor_available`	Determines whether to accept scan requests if no compatible sensors (or compatible versions) are available. The default value is `true`. Also see sensor_version_for_all_jobs. In the following examples, the property is set to `false`: ◦ If a version 24.4 client submits a scan request, and only version 25.2 sensors are available, the scan request is rejected. ◦ If a client submits a request to scan a .NET application and no .NET sensors are available, the scan request is rejected.
`sensor_version_for_all_jobs`	Specifies the version (*<year>.<quarter>* portion only) of the sensor to which the Controller assigns scan jobs for remote translation and scan. For example, if this property is set to 25.2, then scan requests from 24.2, 24.4, or 25.2 version clients are assigned to a 25.2 version sensor. If the OpenText ScanCentral SAST client version is later than the sensor version specified in this property, then the Controller assigns jobs to the sensor version that matches the client version. For example, if this property is set to 24.4, a scan request from a 25.2 version client is assigned to a 25.2 sensor. If this property is not set (default), remote translation and scan jobs are assigned to a sensor with the same version as the OpenText ScanCentral SAST client.
`from_email`	Specifies the outgoing email address that the Controller uses to send job status notifications.
`include_job_status_in_email_subject`	If set to `false`, the job status for the scan request is not included in the email subject of job status notifications. By default, the job status is included in the notification.

`email_allow_list`	Specifies a comma-, colon-, or semicolon-separated list of email domains to which the Controller can send notifications.
	Examples of valid values for this property:
	`*@yourcompanyname.com`
	`*@*yourcompanyname.com`
	`a*@yourcompanyname.com`
	`name1@yourcompanyname.com,name2@yourcompany.com`

`email_deny_list`	Specifies a comma-, colon, or semicolon-separated list of email domains to which the Controller cannot send notifications.
	Examples of valid values for this property:
	`*@yourcompanyname.com`
	`*@*yourcompanyname.com`
	`a*@yourcompanyname.com`
	`name1@yourcompanyname.com,name2@yourcompany.com`

`smtp_host`	Specifies the SMTP server host name.
`smtp_port`	Specifies the SMTP server port number.
`smtp_auth_user`	If your SMTP server requires authentication, uncomment both the `smtp_auth_user` and `smtp_auth_pass` properties and set their values.
`smtp_auth_pass`	You can either use a plain text password or use the `pwtool_keys_file` property to encrypt the password for `smtp_auth_pass`. For information about how to encrypt this password, see Encrypting the Shared Secret on the Controller.
`smtp_ssl`	*If set to `true`, the Controller uses SSL for connections to the SMTP server. By default, the Controller does not use SSL.*
`smtp_ssl_check_trust`	If set to `false`, the SMTP server certificate is always trusted. Otherwise, the certificate trust is based on the certification path (the default)
`smtp_ssl_check_server_identity`	If set to `false`, the SMTP server identity is not checked. Otherwise, the Controller checks server identity as specified by RFC 2595 (the default).
`use_starttls`	If set to `true`, uses the STARTTLS protocol command (opportunistic SSL/TLS) to inform the SMTP server that the email client wants to upgrade from an insecure connection to a secure connection using SSL/TLS. The default value for this property is `false`.
`ssc_lockdown_mode`	If set to `true`, OpenText ScanCentral SAST clients must work with the OpenText ScanCentral SASTController through Application Security. Jobs must be uploaded to an application version and users cannot manually assign scans to specific sensor pools. In SSC lockdown mode, you: ◦ Cannot use the `start` command `-url` option, but must use the `-sscurl` and `-ssctoken` options instead ◦ Must specify the application name and version, or the application version ID, and the `-upload` option when starting the scan ◦ Cannot use the `-pool` option, because the job is automatically assigned to the pool configured for the specified application version
`ssc_ctrl_account_username`	Specifies the user name of a OpenText ScanCentral SAST Controller service account created in Application Security with the **ScanCentral SAST Controller** role. For information about how the Controller uses this account, see Uploading Results to Application Security. For information about how to encrypt this value, see Encrypting the Shared Secret on the Controller.

`ssc_ctrl_account_password`	Specifies the password for the OpenText ScanCentral SAST Controller service account. For information about how to encrypt this value, see Encrypting the Shared Secret on the Controller.
`ssc_remote_ip`	Specifies the remote IP address. You can configure an allowed remote IP address for Application Security. Only requests with a matching remote IP address are allowed.
`ssc_remote_ip_header`	Specifies the remote IP HTTP header, where the Application Security remote IP is found if the `ssc_remote_ip_trusted_proxies_range` property is set. The default value is `X-FORWARDED-FOR`.
`ssc_remote_ip_trusted_proxies_range`	Specifies the remote IP range (in CIDR format). Set this property if Application Security accesses the Controller using a (reverse) proxy server. You can specify comma-separated IP addresses or CIDR network ranges. This is unavailable by default, which means that `ssc_remote_ip_header` is never used to retrieve the remote IP address for Application Security.
`ssc_restapi_connect_timeout`	Specifies the Application Security connection timeout (in milliseconds). The default value is `10000` (or 10 seconds). You can use this, and the `ssc_restapi_read_timeout` property to resolve timeout errors between the Controller and Application Security.
`ssc_restapi_read_timeout`	Specifies the Application Security read timeout (in milliseconds). The default value is `110000` (or 110 seconds). You can use this property and the `ssc_restapi_connect_timeout` property to resolve timeout errors between the Controller and Application Security.
`ssc_scancentral_ctrl_secret`	Specifies the password that Application Security uses to request data from the Controller. Use a string that contains no spaces or backslashes. For instructions on how to encrypt this shared secret value, see Encrypting the Shared Secret on the Controller.

`ssc_url`	Specifies the web address for the Application Security server; all uploads are sent to this address. Examples:
	`https://<ssc_host>:<port>/ssc`
	`https://<ssc_host>:<port>/<context_path>`

`replace_duplicate_scans`	If set to `true`, OpenText ScanCentral SAST replaces a pending scan request with a newer scan request if it is a duplicate. A duplicate scan request occurs if you have more than one scan request that uploads scan results to the same application version in Application Security. The Controller places the new scan request in the same queue position as the one it replaced. Any existing duplicate scan requests with a status of pending are automatically canceled. The scan requests are run sequentially to maintain the submission order. This is typically useful if you submit OpenText ScanCentral SAST scans with upload as part of your build process, which might cause a large queue of unnecessary scan requests that can cause delays for the sensors to process. The default value for this property is `true`. You can override the replacement of duplicate scan requests for specific scans. For more information, see Preventing Replacement of Duplicate Scan Requests.
`ssc_upload_retry_count`	Specifies the maximum number of times the Controller can retry to upload scan results after an upload fails. The default value is `5`. For more information, see Retrying Failed Uploads to Application Security.
`ssc_upload_retry_interval`	Specifies the amount of time (in seconds) the Controller waits after a failed upload before it tries again. The default is 120 seconds (or 2 minutes). For more information, see Retrying Failed Uploads to Application Security.
`swagger_username`	Specifies the user name for access to the OpenText ScanCentral SAST API documentation. For information about how to encrypt this value, see Encrypting the Shared Secret on the Controller.
`swagger_password`	Specifies the password for access to the OpenText ScanCentral SAST API documentation. You can either use a plain text password or use the `pwtool_keys_file` property to encrypt this password. For information about how to encrypt this password, see Encrypting the Shared Secret on the Controller.
`this_url`	Specifies the web address for the Controller; used in emails to refer to this server for manual job result downloads. Example: `https://<controller_host>:8443/scancentral-ctrl`
`worker_auth_token`	Specifies a string that contains no spaces or backslashes to secure the Controller for use by authorized sensors only. If you prefer not to use plain text, you can use an encrypted shared secret as the value for this property. For instructions on how to encrypt this value, see Encrypting the Shared Secret on the Controller.
`worker_inactive_delay`	Specifies the amount of time (in minutes) after which a non-communicating sensor is considered inactive and all jobs are marked as faulted. Assign a value that is much larger than `worker_stale_delay`. Note that this property uses different time units than `worker_stale_delay`. The default value is `60` (or 1 hour).

`worker_stale_delay`	Specifies the amount of time (in seconds) after which a non-communicating sensor is considered inactive. Assign a value that is larger than the `worker_sleep_interval` and `worker_jobwatcher_interval` defined for any sensor. The default value for this property is `60` (or 1 minute).

3. Save and close your `config.properties` file.
4. Start the Controller.

   For instructions, see Starting the Controller.

See also

Installing the Controller

Stopping the Controller

Placing the Controller in maintenance mode

Configuring job cleanup timing on sensors

1.3.4.1. How the Controller assigns scan requests to sensors

The OpenText ScanCentral SAST Controller accepts scan requests and assigns them a sensor of the same version. For example, if a 25.2.0 client submits a scan request, the Controller can assign the job to a version 25.2.0, 25.2.1, or 25.2.2 sensor unless a specific sensor version is specified with the `sensor_version_for_all_jobs` property (see Configuring the Controller).

1.3.4.2. Specifying how the Controller maps scan requests to sensor pools

The `pool_mapping_mode` property in the `config.properties` file determines how the Controller maps scan requests to sensor pools. The valid values for the `pool_mapping_mode` property are:

- `disabled`— In this mode, a OpenText ScanCentral SAST client requests a specific sensor pool when it submits a scan request. Otherwise, the default pool is used.

  For details, see the following table.

- `enabled`— In this mode, if a scan request is associated with an application version in Application Security, the Controller queries Application Security to determine the sensor pool assigned to the application version. Alternatively, a client can request a specific sensor pool when it submits a scan request. A client request for a specific sensor pool takes precedence over a query from the Controller.

  Note

  Sensors in the default sensor pool run scan requests that are not associated with an application version (and no specific pool is requested on the OpenText ScanCentral SAST client command line).

- `enforced`—As with the enabled mode, if a scan request is associated with an application version in Application Security, the Controller queries Application Security for the sensor pool to use for the application version. Otherwise, the Controller targets the default sensor pool for scan requests. A OpenText ScanCentral SAST client cannot request a specific sensor pool in the enforced mode.

  If `ssc_lockdown_mode` is enabled, then the `pool_mapping_mode` is automatically set to enforced and the value set for `pool_mapping_mode` in the `config.properties` file is ignored.

The following table shows how the Application Security integration with OpenText ScanCentral SAST responds to different input when the `pool_mapping_mode` is set to `disabled`, `enabled`, or `enforced`.

Note

By default, in enabled and enforced modes, all application versions are assigned to the default sensor pool.

Input	Disabled	Enabled	Enforced
No pool or version specified	Default sensor pool	Default sensor pool	Default sensor pool
Specific sensor pool (only) specified	Requested sensor pool	Requested sensor pool	Denied
Application version (only) specified	Default sensor pool	SSC-assigned pool	SSC-assigned pool
Invalid sensor pool (only) specified	Denied	Denied	Denied
Invalid application version (only) specified	Denied	Denied	Denied
Valid sensor pool and application version specified	Requested sensor pool	Requested sensor pool	Denied

Invalid sensor pool and valid application version specified	Denied	Denied	Denied
Valid sensor pool but invalid application version specified	Denied	Denied	Denied

See also

Configuring the Controller

1.3.4.3. Encrypting the shared secret on the Controller

Values exist in the Controller configuration file as plain text. You can encrypt the passwords, authentication tokens, and other values for the following properties:

- `client_auth_token`
- `lim_license_pool_password`
- `lim_proxy_password`
- `lim_proxy_user`
- `smtp_auth_pass`
- `ssc_ctrl_account_username`
- `ssc_ctrl_account_password`
- `ssc_scancentral_ctrl_secret`
- `swagger_password`
- `swagger_username`
- `worker_auth_token`

To encrypt a shared secret on the Controller:

1. At the command prompt, type the following:

   ```
   <controller_install_dir>/bin/pwtool <pwtool_keys_file>
   ```

2. When prompted, type the password to encode, and then press **Enter**.

   Note

   For the sake of security, make sure that the pwtool key file you use to encrypt secrets for the Controller is different from the pwtool key file you use to encrypt secrets on sensors.

   The pwtool generates a new key stored in the file on the path specified in step 1 or reuses an existing file on the specified path.

3. Copy the encrypted secret, and paste it as the value for the property you want to encrypt in the `config.properties` file.

   Tip

   OpenText recommends that you assign separate, unique shared secrets for the `client_auth_token`, `smtp_auth_pass`, `ssc_scancentral_ctrl_secret`, and `worker_auth_token` properties.

4. To create additional encrypted shared secrets, repeat steps 1 through 3 for each property value you want to encrypt.
5. Uncommment the following property in the `config.properties` file:

   `pwtool_keys_file=<pwtool_keys_file>`

6. Save and close the `config.properties` file.

See also

Configuring the Controller

1.3.4.4. Configuring the Controller logging

You can configure the Controller logging settings by setting environment variables on the system where you installed the Controller. The following table describes the environment variable settings you can use to configure the Controller logging.

Environment variable name	Default value	Description
CONSOLE_OUT_FILTER_ON_MATCH	`deny`	Specifies the threshold filter for log messages that are written to the console. By default, the setting of `deny` specifies that the log messages are not printed to the console. The valid values for this variable are `deny`, `neutral`, and `accept`. For more information about these values, see the Apache Logging Services Log4j manual for configuring filters.
LOG_FILE_FILTER_ON_MATCH	`neutral`	Specifies the threshold filter for log file messages. By default, the setting of `neutral` specifies that the messages are printed to the log file. Setting this variable to `deny` disables logging to file.
SCANCENTRAL_LOG_LEVEL	`info`	Specifies the log level for OpenText ScanCentral SAST. For more information about log levels, see the Apache Logging Services Log4j website.
SCANCENTRAL_SPRING_LOG_LEVEL	`warn`	Specifies the log level for Spring. For more information about log levels, see the Apache Logging Services Log4j website.
SCANCENTRAL_HIBERNATE_LOG_LEVEL	`warn`	Specifies the log level for Hibernate. For more information about log levels, go to the Apache Logging Services Log4j website.
SCANCENTRAL_LOG_FILE_SIZE	`20MB`	Specifies the maximum size for the log file. If the log file reaches this size, the log file is archived and a new log file is created.
SCANCENTRAL_ARCHIVED_LOGS_NUMBER	`20`	Specifies the maximum number of log files that are archived before new log files start replacing the oldest log files.
SCANCENTRAL_LOGS_FOLDER	`${sys:catalina.base}/logs`	Specifies the location of OpenText ScanCentral SASTController log folder.

See also

1.3.4.5. Avoiding read timeout errors

To avoid read timeout errors that can occur during attempts to upload large log files, you can configure the connection timeout between the Controller and Application Security, between the Controller and sensors, and between the Controller and clients.

To configure the connection timeout between the Controller and Application Security:

1. On the Controller, open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
2. Increase the value of the `ssc_restapi_connect_timeout` and `ssc_restapi_read_timeout` properties to an acceptable threshold (in milliseconds).
3. Save the changes.

To configure the connection timeout between the Controller and a sensor:

1. On the sensor machine, open the `<sast_install_dir>/Core/config/worker.properties` file in a text editor.
2. Uncomment the `restapi_connect_timeout` and `restapi_read_timeout` properties, and then set the value of each to an acceptable threshold (in milliseconds).
3. Save the changes.

To configure the connection timeout between the Controller and a client:

1. On the client machine, open the `<client_install_dir>/Core/config/client.properties` file in a text editor.
2. Uncomment the `restapi_connect_timeout` and `restapi_read_timeout` properties, and then set the value of each to an acceptable threshold (in milliseconds).
3. Save the changes.

See also

Configuring the Controller

Configuring clients

1.3.4.6. Configuring licensing with Fortify License and Infrastructure Manager

OpenText ScanCentral SAST sensors can run OpenText SAST with Fortify License and Infrastructure Manager (LIM). With a LIM managed concurrent license, multiple sensors can share a single license. When a scan job is completed, canceled, times out, or fails, the license is released.

For information about how to set up the LIM with licenses for OpenText SAST, see *OpenText™ Fortify License and Infrastructure Manager Installation and Usage Guide*. To configure the Controller to use LIM, specify the LIM properties as described in Configuring the Controller.

1.3.4.7. Placing multiple standalone clients on the Controller

You can place multiple standalone clients of different supported versions on the Controller. To do this, place any number of client ZIP files for all supported versions into the `<controller_install_dir>/tomcat/client/` directory. You can use any ZIP file names. At startup, the Controller parses the available clients.

To install a patch for a client or sensor version installed on the Controller, place the patch ZIP file into the `<controller_install_dir>/tomcat/client/` directory. If automatic updates is enabled, the clients of that version are automatically updated with the patch. For information about how to enable automatic updates of your clients and sensors, see Enabling Automatic Updates of Clients and Sensors.

1.3.5. Starting the Controller

You can start the OpenText ScanCentral SASTController manually or set it to start automatically, as a service. For information about how to start the Controller automatically, see Installing the Controller as a Service.

To start the Controller manually:

1. To upload your scan results to Application Security, make sure that the Application Security instance is running.
2. On the machine that hosts the Controller, go to `tomcat/bin/`.
3. At the command prompt, run one of the following commands:
    - On a Windows system, run `startup.bat`.
    - On a Linux system, run `./startup.sh`.

   If Tomcat is running as a service, rather than running the startup command, you can just start the service.

See also

Placing the Controller in Maintenance Mode

1.3.6. Placing the Controller in maintenance mode

An abrupt shutdown of the OpenText ScanCentral SASTController can result in the loss of scans already started on sensors. To prevent this from happening, place your Controller in maintenance mode. After you do, the Controller accepts no new job requests from clients and assigns no queued jobs to sensors.

After the Controller is placed in maintenance mode, sensors complete the scans they are currently running, but accept no new scans. After the Controller is back up and running, the sensors again become available.

Tip

If the Controller is in maintenance mode, you can manually shut down any sensor that is not running a scan.

1. Sign in to Application Security as an Administrator and open the **ScanCentral** view.
2. On the navigation pane of the **SAST** page, select **Controller**.
3. Click **START MAINTENANCE MODE**.

The Controller receives the maintenance request from Application Security and, if any sensors are running scans, the Controller mode changes from ACTIVE to WAITING_FOR_JOB_COMPLETED. If no job is being processed, the mode changes directly from ACTIVE to MAINTENANCE. At this point, you can safely shut down the Controller.

See also

Starting the OpenText ScanCentral SAST Controller

Safely shutting down sensors

Removing the Controller from Maintenance Mode

1.3.6.1. Removing the Controller from maintenance mode

To remove the OpenText ScanCentral SASTController from maintenance mode:

1. Sign in to Application Security as an Administrator and open the **ScanCentral** view.
2. On the navigation pane of the **SAST** page, select **Controller**.
3. Click **END MAINTENANCE MODE**.

See also

Placing the Controller in Maintenance Mode

Stopping the Controller

1.3.7. Stopping the Controller

You can stop the OpenText ScanCentral SAST Controller immediately using the following procedure. However, OpenText strongly recommends that you first place the Controller in maintenance mode to preserve any scans that are running.

To stop the Controller:

1. On the machine where the Controller is installed, go to `<controller_install_dir>/tomcat/bin/`:
2. Type one of the following commands:
    - On a Windows system: `shutdown.bat`
    - On a Linux system: `./shutdown.sh`

See also

Placing the Controller in Maintenance Mode

Removing the Controller from Maintenance Mode

Safely shutting down sensors

1.3.8. OpenText ScanCentral SAST API

OpenText ScanCentral SAST provides a RESTful API that enables you perform tasks described in the following table. The tasks are grouped by the grouping in the API Documentation (Swagger UI).

Tasks you can perform	**Request group**
Retrieve the scan requests from the Controller, report job status, and upload artifacts	sensor-controller
Work with scan jobs such as running a new scan or canceling a job	job-controller
Get information from the Controller such as the Application Security URL	info-controller
Check for client or sensor updates	update-controller
Check to see if the Controller is running	core-controller

To use the OpenText ScanCentral SAST API, your application makes an HTTP request and parses the response. The OpenText ScanCentral SAST API uses JSON and XML as its communication format and the standard HTTP methods of GET, POST, and DELETE. URIs have the following structure:

<protocol>://*<controller_url>*/rest/*<api-version>*/*<endpoint>*

The following is an example cURL:

```
curl -X 'GET' \
  'https://my_ctrl_host:8080/scancentral-ctrl/rest/v4/job/a2f0fe34-f810-4c76-8e0b-86dfb4f40c9c/status' \
  -H 'accept: */*' \
  -H 'fortify-client: my_secret'
```

Authentication

Authenticate your API request with a OpenText ScanCentral SAST authentication token. Use the value of the `client_auth_token` or the `worker_auth_token` from the `config.properties` file for the Controller depending on the request. Set the same authentication token in the `fortify-client` header that is set for the `client_auth_token`. Similarly, set the same authentication token in the `fortify-worker` header that is set for `worker_auth_token`. The following table lists which authentication token is used for each request group.

Request group	**client_auth_token**	**worker_auth_token**
sensor-controller		x
job-controller	x	
info-controller	x	x
update-controller	x	x
core-controller	x	

Accessing the OpenText ScanCentral SAST API documentation (Swagger UI)

The documentation describes the input, output, and API endpoints. It also provides the ability to test the endpoints before using them in production.

To access this documentation:

1. Configure the credentials for access to the documentation in the Controller `config.properties` file with the two properties: `swagger_username` and `swagger_password`.

   For more information, see Configuring the Controller.

2. Open a browser and visit *<controller_url>*/rest/swagger-ui/index.html.

Note

OpenAPI documentation in JSON format is available at *<controller_url>*/rest/api-docs.

1.4. About OpenText ScanCentral SAST sensors

OpenText ScanCentral SAST sensors are computers set up to receive scan requests and analyze code using OpenText SAST. A sensor accepts either a project package that contains sources and dependencies, which it translates and scans or it accepts a mobile build session (MBS) file and performs a scan.

For MBS scans, OpenText ScanCentral SAST supports all languages that OpenText SAST supports.

For MBS scans, OpenText ScanCentral SAST uses the OpenText SAST installation specified in the SAST_LOCATION environment variable.

If both OpenText ScanCentral SAST and OpenText SAST are installed in the same directory, you do not need to set the SAST_LOCATION environment variable. The scan will automatically use the OpenText SAST installation found in that location.

If the SAST_LOCATION environment variable is set, OpenText ScanCentral SAST will use the path specified in SAST_LOCATION even if the client is installed in the same directory as OpenText SAST. In this case, the environment variable takes priority.

Tip

Only set the SAST_LOCATION environment variable if OpenText SAST is installed in a different location from the OpenText ScanCentral SAST client.

For remote translation and scans of the projects, your project must be in a language supported for remote translation. For more information, see the *Application Security Software System Requirements* document.

Tip

As you set up your OpenText ScanCentral SAST environment, you can use subnets to segment your build machines from the sensors. The build machines need only communicate with the Controller, which in turn communicates with the sensors.

This section contains the following topics:

1.4.1. Installing sensors

To make it convenient for network administrators to isolate traffic to OpenText ScanCentral SAST sensors, OpenText recommends that you install sensors in a separate subnet. Use the sensors only as scan boxes.

This section contains the following topics:

1.4.1.1. Installing a sensor using OpenText SAST

The following procedure describes how to create a new sensor. For information about how to upgrade an existing sensor, see Upgrading OpenText ScanCentral SAST Sensors.

If you use Windows, you can install the sensor as a Windows service. For instructions, see Installing a Sensor as a Service.

To install a sensor, you must install the OpenText ScanCentral SAST client in addition to the OpenText SAST installation.

To install OpenText ScanCentral SAST client:

1. Obtain OpenText ScanCentral SAST client either from:
   - OpenText ScanCentral SASTController downloaded package from Software Licenses and Downloads (SLD) portal .
   - An existing instance of the OpenText ScanCentral SASTController.
   - `https://tools.fortify.com/scancentral/Fortify_ScanCentral_Client_<version>_x64.zip`, where `<version>` is in the format `<year>.<quarter>.<patch>`. For example, 25.4.0.
2. You can either install OpenText ScanCentral SAST client in the same directory as OpenText SAST or install it in some other directory and set the SAST_LOCATION environment variable:
   - Unzip the contents and copy it to the OpenText SAST installation directory (overwrite the files). Or,
   - Unzip the OpenText ScanCentral SAST client to any directory and set the SAST_LOCATION environment variable with the value of the OpenText SAST installation.

To install a sensor:

1. Use the instructions provided in the *OpenText™ Static Application Security Testing User Guide* to install OpenText SAST.
2. Install the OpenText ScanCentral SAST client (see the Install ScanCentral client procedure above).
3. Open the `<sast_install_dir>/Core/config/worker.properties` file in a text editor.

   Note

   The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST. Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

4. Specify a value for the `worker_auth_token` property.

   If you are using a plain text password, use the password set for the `worker_auth_token` property in the Controller`config.properties` file.

   Note

   Alternatively, you can set the `worker_auth_token` using the `SCANCENTRAL_VM_OPTS` environment variable, like `SCANCENTRAL_VM_OPTS=-Dworker_auth_token=<token>`. For information, see Configuring sensor properties.

   For information about how to generate an encrypted shared secret, see Encrypting the Shared Secret on a Sensor.

5. Save and close your `worker.properties` file.

See also

1.4.1.2. Installing a sensor as a service

If you use Windows services, you can install the OpenText ScanCentral SAST sensor as a Windows service.

To install the sensor as a Windows service:

1. Before running `setupworkerservice.bat` , ensure to either:
    1. Copy the ScanCentral client files into the OpenText SAST installation folder,
       or
    2. Set the SAST_LOCATION environment variable to specify the OpenText SAST installation folder.
2. Go to the `<sc_sast_install_dir>\bin\scancentral-worker-service` directory, and then do one of the following:

   For information about how to encrypt a shared secret, see Encrypting the Shared Secret on a Sensor.

    - To use a plain text password, run the following command:

      ```
      setupworkerservice.bat <sast_version> <controller_url> <shared_secret>
      ```

    - To use an encrypted password, run the following command:

      ```
      setupworkerservice.bat <sast_version> <controller_url> "<encrypted_shared_secret>"
      <path_to_pwtool.keys_file>
      ```

      Important

      Enclose *<encrypted_shared_secret>* in quotes. This ensures that the encrypted shared secret does not get corrupted when the services installer creates the `worker.properties` file.

      where *<sast_version>* is the *<year>.<quarter>* portion of the OpenText SAST version (for example, 25.2).

Caution

The `setupworkerservice` command does not correctly handle `worker_auth_token` tokens that contain the caret character (^). If you must use the caret character as a part of a `worker_auth_token`, use the following formula:

saved_caret_count = carets_used_on_command_line / 8

Examples:

For a `worker_auth_token` that contains a single caret, such as this^that, run the following command:

```
setupworkerservice.bat 25.2 https://url.com this^^^^^^^^that
```

For a `worker_auth_token` that contains two caret characters, such as this^^that, run the following command:

```
setupworkerservice.bat 25.2 https://url.com
this^^^^^^^^^^^^^^^^that
```

3. Start the service, as follows:

```
net start FortifyScanCentralWorkerService
```

The services installer creates the `<sast_install_dir>\Core\config\worker.properties` file for you.

See Next

Enabling Sensor Auto-Start on Windows as a Service

See also

Installing Sensors

1.4.2. Configuring sensors

After you install the OpenText ScanCentral SAST sensors, you can encrypt shared secrets and configure sensor settings such as the connection and read timeouts, sensor expiration time, job cleanup timing, and more.

To configure a sensor:

1. On the sensor machine, open the `<sast_install_dir>/Core/config/worker.properties` file in a text editor.

   Note

   The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST . Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

2. Configure the properties described in the following table.

Sensor property	Description
`worker_auth_token`	Specifies a worker authentication token string that contains no spaces or backslashes to secure the Controller for use by authorized sensors only. Set the same value for the `worker_auth_token` property that you set for the `worker_auth_token` property on the Controller. If you prefer not to use plain text, you can use an encrypted shared secret as the value for this property. For instructions on how to encrypt a shared secret, see Encrypting the shared secret on a sensor.
`pwtool_keys_file`	Specifies the path to a file with pwtool keys. If encrypted passwords are used, this must specify a file with the pwtool keys used to encrypt the passwords. For more information, see Encrypting the shared secret on a client.
`jobs_dir`	Specifies the directory where the job files are created. For more information about customizing persistence for generating job files, see Configuring where to generate job files and the worker_persist.properties file.
`props_dir`	Specifies where to save the `worker_persist.properties` file. For more information, see Configuring where to generate job files and the worker_persist.properties file.
`delete_sca_build_dir`	Specifies whether to delete the temporary working directory after a scan is complete. This temporary directory is used to unpack the project package and store temporary files. For more information, see Preserving the OpenText SAST project root directory.
`restapi_connect_timeout`	Specifies the Controller connection timeout (in milliseconds). The default value is `10000` (or 10 seconds). You can use this, and the `restapi_read_timeout` property to resolve timeout errors between the Controller and the sensor.
`restapi_read_timeout`	Specifies the Controller read timeout (in milliseconds). The default value is `30000` (or 30 seconds). You can use this, and the `restapi_connect_timeout` property to resolve timeout errors between the Controller and the sensor.
`worker_cleanup_age`	Specifies the age (in hours) job files must be before they are removed from the sensor working directory. For more information, see Configuring job cleanup timing on sensors.

`worker_cleanup_interval`	Specifies the frequency (in hours) with which the cleanup process runs. For more information, see Configuring job cleanup timing on sensors.

See also

Configuring proxies for clients and sensors

This section contains the following topics:

- Configuring sensor properties
- Encrypting the shared secret on a sensor
- Configuring the sensor logging
- Setting the maximum run time for scans
- Changing sensor expiration time
- Configuring sensors for remote translation of .NET languages
- Configuring sensors to use the progress command when starting on Java
- Configuring where to generate job files and the worker_persist.properties file
- Configuring job cleanup timing on sensors

1.4.2.1. Configuring sensor properties

In addition to setting sensor properties in the `<sast_install_dir>/Core/config/worker.properties` file, you can add them to the SCANCENTRAL_VM_OPTS environment variable. A property value set in the SCANCENTRAL_VM_OPTS environment variable overrides any equivalent property set in the `worker.properties` file. The following example sets the sensor authorization token and the connection timeout between the Controller and a sensor:

Note

The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST. Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

- On a Windows system: `set SCANCENTRAL_VM_OPTS=-Dworker_auth_token=<token> -Drestapi_connect_timeout=10000`
- On a Linux system: `export SCANCENTRAL_VM_OPTS=-Dworker_auth_token=<token> -Drestapi_connect_timeout=10000`

Note

You can also set Java system properties (such as `-Djava.io.tmpdir=<path>`) in the SCANCENTRAL_VM_OPTS environment variable.

1.4.2.2. Encrypting the shared secret on a sensor

Values exist in the OpenText ScanCentral SAST sensor configuration file as plain text. You can encrypt the `worker_auth_token` property value.

To encrypt a shared secret on a sensor:

1. At the command prompt, run the following command:

   ```
   <sast_install_dir>/bin/pwtool <pwtool_keys_file>
   ```

2. When prompted, type the password to encode, and then press **Enter**.

   Note

   For the sake of security, make sure that the pwtool key file you use to encrypt secrets for sensors is different from the pwtool key file you use to encrypt secrets on the Controller.

   The pwtool generates a new `pwtool.keys` file to *`<pwtool_keys_file>`* and prints a new encrypted secret to the console.

3. Open the `worker.properties` file in a text editor and update the values for the following properties:

   1. Copy the encrypted secret and paste it as the value for `worker_auth_token` property.
   2. Add the name of your pwtool keys file:

      `pwtool_keys_file=`*`<pwtool_keys_file>`*

4. Save and close the `worker.properties` file.

See also

Installing Sensors

1.4.2.3. Configuring the sensor logging

You can configure the sensor logging settings by setting environment variables on the system where the sensors are installed. The following table describes the environment variable settings that you can use to configure the sensor logging.

Environment variable name	Default value	Description
CONSOLE_OUT_FILTER_ON_MATCH	`deny`	Specifies the threshold filter for log messages that are written to the console. By default, the setting of `deny` specifies that the log messages are not printed to the console. The valid values for this variable are `deny`, `neutral`, and `accept`. For more information about these values, see the Apache Logging Services Log4j manual for configuring filters.

LOG_FILE_FILTER_ON_MATCH	`neutral`	Specifies the threshold filter for log file messages. By default, the setting of `neutral` specifies that the messages are printed to the log file. Setting this variable to `deny` disables logging to file.
SCANCENTRAL_LOG_LEVEL	`info`	Specifies the log level for OpenText ScanCentral SAST. For more information about log levels, see the Apache Logging Services Log4j website.
SCANCENTRAL_SPRING_LOG_LEVEL	`warn`	Specifies the log level for Spring. For more information about log levels, see the Apache Logging Services Log4j website.
SCANCENTRAL_LOG	Windows: `<user_app_data>\Local\Fortify\scancentral-<version>\log` Linux: `<user_home>\.fortify\scancentral-<version>\log`	Specifies the location of Fortify ScanCentral log.

Configuring the external logger configuration file

To configure an external logger configuration file, set the default log configuration file `-Dlog4j2.configurationFile` using the `SCANCENTRAL_VM_OPTS` environmental variable. Both SC SAST Launcher and Client take the new configuration file from the `SCANCENTRAL_VM_OPTS` environmental variable instead of the default one.

For example, `SCANCENTRAL_VM_OPTS=-Dlog4j2.configurationFile=file:///C:/temp/log4j2.xml`**.** The specified configuration file is `C:/temp/log4j2.xml`.

Ensure to specify a protocol, like `file:///`.

1.4.2.4. Setting the maximum run time for scans

By default, a sensor can run a scan for an indefinite period of time, which prevents it from running other scans. You can limit the amount of time scans can run on sensors for a specific job, for a specific sensor, or globally for all sensors.

The following precedence rules apply to timeout settings:

- Job timeout settings override any sensor-specific or global timeout settings.
- Sensor timeout configured on the command line overrides a global timeout setting.

Configuring the maximum run time for a specific job

To configure the maximum run time of one minute for a specific job, run the following command:

```
scancentral -url <controller_url> start --scan-timeout 1
```

To configure the maximum run time of two minutes for a specific sensor, run the following:

```
scancentral -url <controller_url> worker --scan-timeout 2
```

Configuring the maximum run time for all sensors

To configure the maximum run time for all sensors:

1. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
2. Set the `scan_timeout` property value to the maximum number of minutes for scans to run on sensors.
3. Save and close the `config.properties` file.

1.4.2.5. Changing sensor expiration time

By default, sensors expire 168 hours after they become inactive. To change this default value:

1. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
2. Set the `worker_expiry_delay` property value to the number of hours to elapse after inactivity before sensors expire.
3. Save and close the `config.properties` file.

1.4.2.6. Configuring sensors for remote translation of .NET languages

To use your OpenText ScanCentral SAST sensors for remote translation of code written in a .NET language, configure at least one sensor with the software required to support .NET. Sensors on Windows or Linux can accept any package for remote translation built by MSBuild and dotnet if .NET capability is enabled. See the *Application Security Software System Requirements* document for specific .NET version requirements.

After you start a sensor, it automatically detects if a supported version of .NET is installed and displays a message that .NET capability is enabled. This indicates that the sensor can now translate .NET projects.

Important

To avoid Windows errors caused by too long a path during a .NET translation, start sensors from a directory with a short name and path.

See also

Installing sensors

Starting the sensors

1.4.2.7. Configuring sensors to use the progress command when starting on Java

To use the `progress` command to check the progress of your OpenText SAST scans, you must complete the following sensor configuration:

1. Create a JMX access file, and add the following text to it:

   ```
   <user_role> readonly
   ```

   where `<user_role>` is text that represents something like a user name.

2. Create a JMX password file, and add the following text to it:

   ```
   <user_role> <password> readonly
   ```

   where `<user_role>` is the value you specified in the JMX access file.

3. Run one of the following commands:

   - On a Windows system, `cacls jmxremote.password /P <username>:R`
   - On a Linux system, `chmod 600 jmxremote.password`

4. Open the `worker.properties` file in a text editor, and then add the following properties to it:

   ```
   sca_jmx_port=<port> sca_jmx_access_file=<path_to_access_file> sca_jmx_password_file=<path_to_password_file> sca_jmx_password=<password> sca_jmx_user=<user_role> sca_jmx_auth=true
   ```

5. Save and close the `worker.properties` file.

After you complete this configuration, OpenText ScanCentral SAST clients start on the specified port using JMX password authentication. Make sure that the port is not already bound.

Caution

If you use `sca_jmx_auth`, you can start only one sensor. Any attempt to open a new OpenText SAST instance results in a bind port error. To have multiple sensors on a machine, you must have several OpenText ScanCentral SAST instances, each with its own `worker.properties` file.

1.4.2.8. Configuring where to generate job files and the worker_persist.properties file

For containerized deployments, it is useful to determine where files are generated so that you can customize persistence. This enables you to persist the `worker_persist.properties` file, which you need to maintain sensor pool assignments, without having to keep all the old job files.

Note

If you choose not to configure these locations, the default locations are used. The default location for the `worker_persist.properties` file is `<working_dir>/props`. The default location for the job files is `<working_dir>/jobs`.

To configure where job files and the `worker_persist.properties` file are generated:

1. On a sensor machine, open the `<sast_install_dir>/Core/config/worker.properties` file in a text editor.

   Note

   The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST. Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

2. Specify directories for the following properties:
   - The `props_dir` property specifies where the `worker_persist.properties` file is saved.
   - The `jobs_dir` property specifies the directory where the job files are created.
3. Save and close your `worker.properties` file.
4. Restart the sensor.

1.4.2.9. Configuring job cleanup timing on sensors

To prevent the progressive loss of disc space as job files accumulate, OpenText ScanCentral SAST sensors automatically clean up internal job files (packages received from the Controller, FPR files, logs, and so on), and OpenText SAST build files related to cleaned OpenText ScanCentral SAST jobs. Although you cannot turn off this feature, you can configure its timing.

To configure the timing of job file cleanup on a sensor:

1. Open the `<sast_install_dir>/Core/config/worker.properties` file in a text editor.

   Note

   The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST. Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

2. Configure the following properties based on your scheduling needs.

Property	Description	Default value
`worker_cleanup_age`	Age (in hours) job files must be before they are removed from the sensor working directory	168 (one week)
`worker_cleanup_interval`	Frequency (in hours) with which the cleanup process runs	1

3. Save and close your `worker.properties` file.
4. Restart the sensor.

1.4.3. Starting the sensors

To start the OpenText ScanCentral SAST sensors:

1. Start the Controller if it is not already running.
2. On each sensor, go to the `<sast_install_dir>/bin/` directory.

Note

The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST. Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

3. Start the sensor by typing the following command:

```
scancentral –url <controller_url> worker
```

If the sensor starts successfully, it displays messages to signal its waiting status on the console. After you verify that the sensor is working, you can create a Startup Task in Windows Task Scheduler or add it to your startup scripts. For more information, see Configuring Sensor Auto-Start.

Note

Make sure that you run each sensor consistently from the same directory. Otherwise, its UUID changes and, if OpenText ScanCentral SAST is connected to Application Security, Application Security identifies it as different sensor.

See also

Placing the Controller in Maintenance Mode

Configuring Sensor Auto-Start

1.4.3.1. Configuring sensor auto-start

The following topics provide general guidance to enable sensor auto-start and might not be appropriate in all environments. OpenText strongly recommends that you review the instructions with your system administrator and make any changes required for your environment.

This section contains the following topics:

1.4.3.1.1. Enabling sensor auto-start on Windows as a service

Make sure the OpenText ScanCentral SAST Controller is running before you perform the following procedure.

To enable sensor auto-start on Windows as a service:

1. Log in to the sensor machine as a local user with administrator permissions.

   Sensors are dedicated machines intended only to run OpenText SAST on behalf of OpenText ScanCentral SAST. Do not share them with any other service. To avoid issues associated with insufficient permissions, use a fully privileged administrator account for the auto-start setup.

2. Open a command prompt and go to the `<sast_install_dir>\bin\scancentral-worker-service/` directory.

   Note

   The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST. Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

3. Run `setupworkerservice.bat` with no options to display the usage help.
4. Re-run the batch script with the required options included.
5. Open Windows Services and check to make sure that the sensor service is present.
6. Right-click the listed sensor service, and then select **Start**.
7. OpenText recommends that you change the startup type setting to **Manual** until you verify that the sensor runs successfully. After verification, change the startup type setting to **Automatic (Delayed Start)** in Windows Services.
8. Make sure that the sensor communicates with the Controller.

See also

Installing a Sensor as a Service

Troubleshooting Sensor Auto-Start as a Windows Service

1.4.3.1.2. Enabling sensor auto-start on Windows as a scheduled task

To enable OpenText ScanCentral SAST sensor auto-start on Windows as a scheduled task:

1. Log on to the sensor machine as a local user with administrator permissions.

   Sensors are dedicated machines intended only to run OpenText SAST on behalf of OpenText ScanCentral SAST. Do not share them with any other service. To avoid issues associated with insufficient permissions, use a fully privileged administrator account for the auto-start setup.

2. Start the Task Scheduler.
3. In the **Actions** pane, select **Create Task**.
4. On the **General** tab, provide the following information:
   1. In the **Name** box, type a name for the task.
   2. Click **Run whether user is logged on or not**.
5. Click the **Actions** tab, and then click **New**.

   The New Action dialog box opens.

   1. In the **Action** list, select **Start a program**.
   2. In the **Program/script** box, type the directory path to your `scancentral.bat` file (for example, `<sast_install_dir>\bin\scancentral.bat`).

      Note

      The `<sast_install_dir>` is only valid if the user installs the OpenText ScanCentral SAST client in the same directory as the OpenText SAST. Otherwise, this `worker.properties` file is under the OpenText ScanCentral SAST client installation directory.

   3. In the **Add arguments (optional)** box, type the following:

      ```
      –url https://<host>:<port>/scancentral-ctrl worker >taskout.txt 2>&1
      ```

   4. In the **Start in (optional)** box, type the path to the OpenText ScanCentral SAST sensor `bin` directory (for example, `<sast_install_dir>\bin\`).
   5. Click **OK**.
6. Click the **Triggers** tab.
7. Make sure that the **At startup** trigger is enabled, and then click **OK**.
8. Click the **Settings** tab.
9. Make sure the **Stop the task if it runs longer than** check box is cleared, and then click **OK**.
10. Restart the machine.

The script output in the `taskout.txt` file indicates whether the sensor started successfully.

You can also start and stop the scheduled task manually from the Task Scheduler when you are logged into the machine.

1.4.3.1.3. Enabling sensor auto-start on a Linux system

The following procedure has been tested with Red Hat® Enterprise Linux®; there might be some variation for other Linux varieties. Review these steps with your system administrator before you make any changes.

To enable OpenText ScanCentral SAST sensor auto-start on a Linux system:

1. Log in to the machine as “root.”
2. Run the `visudo` command to edit the `sudoers` file and disable requiretty.

```
Defaults !requiretty
```

Note

You can also disable requiretty per user.

3. Create a systemd service unit file:
    1. Open a new file for the service:

```
vi /etc/systemd/system/scancentral-sensor.service
```

    2. Add the following content, and modify the paths, token, and username as needed:

```
[Unit]
Description=Fortify ScanCentral SAST Sensor

[Service]
User=<username>
Environment="SAST_LOCATION=<sast_install_dir>" "SCANCENTRAL_VM_OPTS=-
Dworker_auth_token=<worker authentication token>"
ExecStart=<client_install_dir>/bin/scancentral -url <controller_url> worker
StandardOutput=append:<client_install_dir>/bin/workerout.txt
StandardError=append:<client_install_dir>/bin/workerout.txt
WorkingDirectory=<working_dir>
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Note

- The <working_dir> should be a directory that the <username> has write permissions to.
- The SCANCENTRAL_VM_OPTS environment variable can be used to set additional sensor properties. For more information, see Configuring sensor properties.

4. Reload the systemd and enable the service:

```
systemctl daemon-reload
systemctl enable scancentral-sensor
systemctl start scancentral-sensor
```

5. Check the setup:
    1. Reboot and log in to the machine as "root".
    2. To verify the processes under root, type:

        ```
        ps -x | grep java
        ```

        Verify that the output shows that the sensor is not started under root.
    3. To verify the processes under the user, type:

        ```
        sudo -u <username> ps x | grep java
        ```

        Verify that the output displays the sensor process.
    4. To verify the existence and contents of the script output file, type:

        ```
        tail -f <client_install_dir>/bin/workerout.txt
        ```

        Note

        The file /bin/workerout.txt must match what was configured for the StandardOutput and StandardError in the scancentral-sensor.service

        For example:

        ```
        tail -f
        /home/<username>/Fortify/Fortify_ScanCentral_Client_25.4.0_x64/bin/workerout.txt
        ```

1.4.4. Safely shutting down sensors

This topic describes how to safely shutdown OpenText ScanCentral SAST sensors from Application Security.

Important

If the Controller is in maintenance mode, you cannot shut down sensors from Application Security.

To shut down active sensors:

1. Sign in to Application Security as an Administrator and select the **ScanCentral** view.
2. On the navigation pane of the **SAST** page, select **Sensors**.
3. In the sensors table, do one of the following:
    - Expand the row for a sensor you want to shut down, and then click **SHUT DOWN**.
    - Select the check boxes for one or more sensors you want to shut down, and then click **SHUT DOWN**.

    If the **SHUT DOWN** button is not enabled, it can mean that:
    - The Controller is in maintenance mode.
    - The sensor is already shut down.
    - The sensor is inactive or disabled.

If a sensor you shut down is running a scan, the **State** value for the sensor changes from **Active** to **Shutdown scheduled**. After the scan is complete, the state then changes to **Inactive**.

1.5. About OpenText ScanCentral SAST clients

A OpenText ScanCentral SAST client can generate packages with sources and dependencies, which are uploaded to the Controller for remote translation and scan on sensors. You can use this functionality independent of OpenText SAST.

A OpenText ScanCentral SAST client can also run on a build machine where OpenText SAST translates code and generates mobile build sessions (MBS). The translated source code, along with optional and required data, such as custom rules and OpenText SAST command-line options, are uploaded to the Controller for analysis by sensors.

This section contains the following topics:

1.5.1. Embedded clients and standalone clients

A client can be either an *embedded* client, which is configured to work with an OpenText SAST distribution or a *standalone* client, which is independent of OpenText SAST.

An *embedded* client is an instance of the OpenText ScanCentral SAST client which is configured to work with an instance of OpenText SAST. This can be either:

- A OpenText ScanCentral SAST client that has been installed in the same installation directory as OpenText SAST , or
- A OpenText ScanCentral SAST client that has been configured to work with an OpenText SAST installation by setting the SAST_LOCATION environment variable. The SAST_LOCATION environment variable should be set to the installation directory of OpenText SAST.

The interface for issuing OpenText ScanCentral SAST commands is installed on your client. You use this interface to set the options for the scan and communicate your intentions to the Controller.

The files used to create OpenText ScanCentral SAST sensors and embedded clients are the same. The only difference is how you invoke the functionality from the command line. To use OpenText ScanCentral SAST as a sensor, you run OpenText ScanCentral SAST using the `worker` command. To use OpenText ScanCentral SAST as an embedded client to start a scan, invoke it using the `start` command. Sensor functionality depends on OpenText SAST. So, you can have a standalone client, but not a standalone sensor. You can use an embedded client for either local translation and remote scan or remote translation and scan.

A standalone client does not require the installation of OpenText SAST. You can use it to create a package of the code with its dependencies to send to the Controller for remote translation and scan.

1.5.2. OpenText SAST and OpenText ScanCentral SAST version compatibility

The OpenText SAST version on a OpenText ScanCentral SAST client must be compatible with the OpenText SAST version installed on the sensors. The version number format is *<year>.<quarter>.<patch>.<buildnumber>* (for example 25.4.0.0068). By default, the *<year>* and *<quarter>* portions of the OpenText SAST version numbers on both the client and sensor must match. For example, version 25.2.0 works with version 25.2.1. For other options of supported version compatibility, see the Controller configuration property `sensor_version_for_all_jobs` in Configuring the Controller.

To determine the OpenText SAST version, run the command `sourceanalyzer -version`.

1.5.3. Installing clients

Unless you use a language that supports offloading the translation phase of analysis to your sensors, you must have a licensed copy of OpenText SAST on each machine you plan to use as OpenText ScanCentral SAST clients. If you use a language supported for remote translation, you can install standalone clients, independent of OpenText SAST. For a list of languages that OpenText ScanCentral SAST supports, see the *Application Security Software System Requirements* document.

In this help, *<client_install_dir>* refers to the OpenText ScanCentral SAST client installation directory.

See also

OpenText SAST and OpenText ScanCentral SAST version compatibility

Installing a Standalone Client

1.5.3.1. Installing a standalone client

To submit scan requests for remote translation and remote scan to your OpenText ScanCentral SAST sensors, you can use standalone clients. A standalone client is independent of an OpenText SAST installation.

To install a standalone client:

1. Copy the OpenText ScanCentral SAST client files to your machine by doing one of the following:
   - Install from a OpenText ScanCentral SAST client ZIP file:
     1. Extract the contents of the `Fortify_ScanCentral_Client_<version>_x64.zip` file to any directory on your machine.

        Important

        Make sure that the installation path contains no spaces.

     2. On Linux systems, give the `scancentral`, `pwtool`, and `packagescanner` files execute permission.
     3. Add `<client_install_dir>/bin` to your PATH environment variable.

        The *<client_install_dir>* is the directory where you extracted the OpenText ScanCentral SAST client ZIP.
     4. Set the `SCANCENTAL_JAVA_HOME` environment variable to point to a Java version that OpenText ScanCentral SAST client supports, and make sure that you add the Java executable to the `PATH` environment variable.
   - Install the OpenText ScanCentral SAST client as a component of an OpenText™ Application Security Tools installation.

     For instructions, see the *OpenText™ Application Security Tools Guide*.
2. Open the `<client_install_dir>/Core/config/client.properties` file in a text editor.
3. Set the same value for the `client_auth_token` property that you set for the `client_auth_token` property on the Controller (in the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file). For information about how to generate an encrypted shared secret, see Encrypting the Shared Secret on a Client.

   Note

   You can also set the `client_auth_token` using the `SCANCENTRAL_VM_OPTS` environment variable, like `SCANCENTRAL_VM_OPTS=-Dclient_auth_token= <token>`. For information, see Configuring client properties.
4. Save and close the `client.properties` file.

See also

Placing Multiple Standalone Clients Under the Controller

Configuring client properties

Upgrading a Client

1.5.4. Configuring clients

After you install the OpenText ScanCentral SAST client, you can encrypt shared secrets and configure client settings such as connection and read timeouts, proxy settings, and more.

To configure the OpenText ScanCentral SAST client:

1. On the client machine, open the `<client_install_dir>/Core/config/client.properties` file in a text editor.
2. Configure the properties described in the following table.

Client property	Description
`client_auth_token`	Specifies a client authentication token string that contains no spaces or backslashes to secure the Controller for use by authorized clients only. Set the same value for the `client_auth_token` property that you set for the `client_auth_token` property on the Controller. If you prefer not to use plain text, you can use an encrypted shared secret as the value for this property. For instructions on how to encrypt a shared secret, see Encrypting the shared secret on a client.
`pwtool_keys_file`	Specifies the path to a file with pwtool keys. If encrypted passwords are used, this must specify a file with the pwtool keys used to encrypt the passwords. For more information, see Encrypting the shared secret on a client.
`restapi_connect_timeout`	Specifies the Controller connection timeout (in milliseconds). The default value is `10000` (or 10 seconds). You can use this, and the `restapi_read_timeout` property to resolve timeout errors between the Controller and the client.
`restapi_read_timeout`	Specifies the Controller read timeout (in milliseconds). The default value is `30000` (or 30 seconds). You can use this, and the `restapi_connect_timeout` property to resolve timeout errors between the Controller and the client.
`use_system_gradle`	If set to `true`, OpenText ScanCentral SAST uses the Gradle included in the PATH environment variable. By default, OpenText ScanCentral SAST uses the Gradle wrapper included in the project being analyzed.
`debricked_cli_dir`	(OpenText Core Application Security users only) Specifies a custom location for the Debricked CLI installation.

   For a description of the proxy-related properties for clients, see Configuring proxies for clients and sensors.
3. Save and close the `client.properties` file.

See also

Configuring proxies for clients and sensors

Configuring the Controller

This section contains the following topics:

- Configuring client properties
- Encrypting the shared secret on a client
- Configuring the client logging
- Configuring proxies for clients and sensors

1.5.4.1. Configuring client properties

In addition to setting client properties in the `<client_install_dir>/Core/config/client.properties` file, you can add them to the SCANCENTRAL_VM_OPTS environment variable. A property value set in the SCANCENTRAL_VM_OPTS environment variable overrides any equivalent property set in the `client.properties` file. The following example sets the client authorization token and the connection timeout between the Controller and a client:

- On a Windows system: `set SCANCENTRAL_VM_OPTS=-Dclient_auth_token=<token> -Drestapi_connect_timeout=10000`
- On a Linux system: `export SCANCENTRAL_VM_OPTS=-Dclient_auth_token=<token> -Drestapi_connect_timeout=10000`

Note

You can also set Java system properties (such as `-Djava.io.tmpdir=<path>`) in the SCANCENTRAL_VM_OPTS environment variable.

1.5.4.2. Encrypting the shared secret on a client

Passwords exist in the OpenText ScanCentral SAST client configuration file as plain text. You can encrypt the `client_auth_token` property value.

To encrypt a shared secret on a client:

1. At the command prompt, run one of the following commands:
   - For an embedded client installed with OpenText SAST, run:

     ```
     <sast_install_dir>/bin/pwtool <pwtool_keys_file>
     ```
   - For a standalone client, run:

     ```
     <client_install_dir>/bin/pwtool <pwtool_keys_file>
     ```
2. When prompted, type the password to encode, and then press **Enter**.

   The pwtool generates a new key in the file on the specified path or reuses an existing file and prints the encrypted password.
3. Open the `client.properties` file in a text editor and update the values for the following properties:
   1. Copy the new encrypted secret, and paste it as the value for the `client_auth_token` property.
   2. Add the name of your pwtool keys file:

      `pwtool_keys_file=<pwtool_keys_file>`
4. Save and close the `client.properties` file.

See also

Installing clients

1.5.4.3. Configuring the client logging

You can configure the client logging settings by setting environment variables on the system where the client is installed. The following table describes the environment variable settings that you can use to configure the client logging.

Environment variable name	Default value	Description
CONSOLE_OUT_FILTER_ON_MATCH	`deny`	Specifies the threshold filter for log messages that are written to the console. By default, the setting of `deny` specifies that the log messages are not printed to the console. The valid values for this variable are `deny`, `neutral`, and `accept`. For more information about these values, see the Apache Logging Services Log4j manual for configuring filters.

LOG_FILE_FILTER_ON_MATCH	`neutral`	Specifies the threshold filter for log file messages. By default, the setting of `neutral` specifies that the messages are printed to the log file. Setting this variable to `deny` disables logging to file.
SCANCENTRAL_LOG_LEVEL	`info`	Specifies the log level for OpenText ScanCentral SAST. For more information about log levels, see the Apache Logging Services Log4j website.
SCANCENTRAL_SPRING_LOG_LEVEL	`warn`	Specifies the log level for Spring. For more information about log levels, see the Apache Logging Services Log4j website.
SCANCENTRAL_LOG	Windows: `<user_app_data>\Local\Fortify\scancentral-<version>\log` Linux: `<user_home>\.fortify\scancentral-<version>\log`	Specifies the location of Fortify ScanCentral log.

Configuring the external logger configuration file

To configure an external logger configuration file, set the default log configuration file `-Dlog4j2.configurationFile` using the `SCANCENTRAL_VM_OPTS` environmental variable. Both SC SAST Launcher and Client take the new configuration file from the `SCANCENTRAL_VM_OPTS` environmental variable instead of the default one.

For example, `SCANCENTRAL_VM_OPTS=-Dlog4j2.configurationFile=file:///C:/temp/log4j2.xml`**.** The specified configuration file is `C:/temp/log4j2.xml`.

Ensure to specify a protocol, like `file:///`.

1.5.4.4. Configuring proxies for clients and sensors

If all your outbound traffic must go through a proxy, you can configure one for your OpenText ScanCentral SAST clients.

To configure proxies for clients and sensors:

1. Go to the `<client_install_dir>/Core/config/` directory, and, in both the `client.properties` and `worker.properties` files, uncomment, and then set values for the properties listed in the following table.

Property	Description
ctrl_proxy_host	Type the name of the Controller proxy host.
ctrl_proxy_port	Type the Controller proxy port number.
ctrl_proxy_user	If authentication is required, type a user name.
ctrl_proxy_password	If authentication is required, type the password for the proxy user.
ssc_proxy_host	Type the name of the Application Security proxy host.
ssc_proxy_port	Type the number of the Application Security proxy port.
ssc_proxy_user	If authentication is required, type the proxy user name.
ssc_proxy_password	If authentication is required, type the password for the proxy user.

2. Save and close the `client.properties` and `worker.properties` files.
3. To enable proxy authentication when the Controller is running under HTTPS, add the `-Djdk.http.auth.tunneling.disabledSchemes` Java property to the SCANCENTRAL_VM_OPTS environment variable by typing one of the following commands:
    - On a Windows system: `set SCANCENTRAL_VM_OPTS=-Djdk.http.auth.tunneling.disabledSchemes`
    - On a Linux system: `export SCANCENTRAL_VM_OPTS=-Djdk.http.auth.tunneling.disabledSchemes`

1.6. Upgrading OpenText ScanCentral SAST components

OpenText ScanCentral SAST-related functionality in Application Security requires updated OpenText ScanCentral SAST components.

Important

It is recommended to:

- Upgrade the Controller before you upgrade the OpenText ScanCentral SAST sensors and clients.
- Match the Controller version to your Application Security version.

This section contains the following topics:

1.6.1. Supporting multiple OpenText SAST versions

To support heterogeneous environments and facilitate phased OpenText SAST upgrades, the Controller supports scan request routing based on the OpenText SAST version. For example, you can configure two different client machines, each with a different OpenText SAST version, and configure the sensors with compatible OpenText SAST versions. By default, jobs from each client are then routed to the sensor that has the same OpenText SAST version installed. You can change this behavior and specify a specific sensor version for all jobs (see Configuring the Controller). You can also specify the OpenText SAST version to send the job using the `-sastver` option (see Start command).

If you have an existing OpenText SAST installation (that includes the OpenText ScanCentral SAST client executable file in your path and a mixed version environment, make sure that you are running the latest OpenText ScanCentral SAST executable when you run the client and sensor commands. (Use explicit paths.) To add capacity (new clients or sensors), you can clone the VMs you have already configured or use sensor hosts with the same specifications and installation directory structure.

Important

If you clone VMs, then after cloning, you *must* remove the `worker_persist.properties` file from the directory specified for the `props_dir` property (see Configuring Where to Generate Job Files and the worker-persistence.properties File).

Use sensor machines dedicated to OpenText ScanCentral SAST and run sensors under a dedicated user name. Run only one sensor instance per machine.

If the Controller and Application Security run on different machines, make sure that the `ssc_url` and `this_url` properties in the `scancentral-ctrl/WEB-INF/classes/config.properties` file, and the Controller URL set on Application Security (select **Administration > Configuration > ScanCentral SAST**) resolve to the correct IP addresses.

Make sure a security system or other tool does not block the following channels of communication:

- Controller to Application Security port (for uploads of scan results)
- Application Security to the OpenText ScanCentral SAST Controller port (for OpenText ScanCentral SAST administration console functionality)
- Clients to the Controller port
- Sensors to the Controller port
- Clients to the Application Security port (required only if Application Security is in lockdown mode, or if you use the `-sscurl` option)

1.6.2. Upgrading the Controller

To upgrade your OpenText ScanCentral SASTController:

1. (Recommended) Allow all jobs to finish.

   Place the Controller in maintenance mode so that sensors complete all currently running scans.

2. Shut down the Controller.
3. Back up the existing Controller directories.
4. Install the new Controller in a different location from the existing Controller directories.

   If you plan to install the Controller as a Windows or Linux service, make sure that you install the Controller in a directory where the local service (Windows) or the user or group using the service (Linux) has access.

5. If your existing `config.properties` file has been modified, you must manually apply any changes you made to the new `config.properties` file.

   You cannot simply copy the existing `config.properties` file.

6. If (and only if) you are upgrading your Controller from version 23.1.x to version 25.4.0, run the migration script as follows:

   1. Open a command prompt and go to the new 25.4.0Controller installation directory.
   2. At the command prompt, enter `cd db-migrate`.
   3. Identify the `cloudCtrlDb` and Controller directories for the existing OpenText ScanCentral SAST version. In the following example, the existing Controller is installed on a Windows system in the `C:\scancentral23.1.0` directory:

      ```
      C:\scancentral23.1.0\tomcat\cloudCtrlDb
      C:\scancentral23.1.0\tomcat\webapps\scancentral-ctrl
      ```

   4. Run the following command.

      This command example includes the example directories shown in the preceding step.

      ```
      migrate C:\scancentral23.1.0\tomcat\cloudCtrlDb C:\scancentral23.1.0\tomcat\webapps\scancentral-ctrl
      ```

      The migration script generates the `cloudCtrlDb` directory in the current working directory.

7. Go to the `jobFiles` and `cloudCtrlDb` directories of the existing Controller, and then copy them to the corresponding directories for the new Controller.

   Important

   If you migrated the database (step 6), make sure that you copy the migrated database (`cloudCtrlDb` directory) to the new Controller installation directory.

   The process owner must have write permission for the database file in the `cloudCtrlDb` directory. If you run the OpenText ScanCentral SAST Controller as a Windows service, make sure that the LOCAL SERVICE account has write permission to the database file.

   To change these directories, edit the `job_file_dir` and `db_dir` properties in the `config.properties` file (see Configuring the Controller).

8. Start the new Controller.

   The database is automatically migrated.

9. (Optional) Remove the Controller directories for the previous version.

See also

Installing the Controller

Upgrading OpenText ScanCentral SAST components

Upgrading sensors

Enabling automatic updates of clients and sensors

1.6.3. Upgrading sensors

Important

If OpenText SAST is installed in a location that requires that you have administrator permissions to modify it (for example in `Program Files`), then to update a sensor you must start it with administrator permissions. Otherwise, the sensor cannot write files to disk. If automatic updates is enabled, major updates on standalone clients must finish successfully before the sensor can start. With automatic updates enabled, patch updates allow sensors and clients to start unless the upgrade fails.

To upgrade your OpenText ScanCentral SAST sensors (on Windows or Linux), you can either install the latest version of OpenText SAST, or unzip the `Fortify_ScanCentral_Client_<version>_x64.zip` file. You can use the client-only approach if you plan only to use remote translation and analysis workflows. Local translation requires a local OpenText SAST installation. You can also find the OpenText ScanCentral SAST client inside the `Fortify_ScanCentral_Controller_<version>_x64.zip` file in the `tomcat/client/scancentral.zip` directory.

Tip

You can configure automatic upgrades of both sensors and clients. For details, see Enabling automatic updates of clients and sensors.

To upgrade sensors, install a new version of OpenText SAST and a new version of OpenText ScanCentral SAST client, and set the SAST_LOCATION environment variable.

See also

Enabling automatic updates of clients and sensors

Starting the Sensors

Configuring Sensors to Use the Progress Command when Starting on Java

Upgrading the Controller

1.6.4. Upgrading a client

Important

OpenText recommends that your standalone OpenText ScanCentral SAST clients and your OpenText SAST installation be the same version.

To upgrade a standalone client (independent of OpenText SAST), do one of the following:

- Delete the existing client, and then extract the `Fortify_ScanCentral_Client_<version>_x64.zip` file to any directory on the machine.
- Extract the contents of the `Fortify_ScanCentral_Client_<version>_x64.zip` file on top of the existing client.

The procedure to upgrade an embedded client is the same as upgrading a sensor (see Upgrading sensors).

See also

Creating a Standalone Client

1.6.5. Enabling automatic updates of clients and sensors

You can have all OpenText ScanCentral SAST clients and sensors check with the Controller after a manual update and following each startup to determine whether updates are available (meaning the client or sensor version is earlier than the Controller version). Then, if an update is available, the Controller updates all sensors and clients.

The upgrade paths for clients and sensors are as follows:

- You can update standalone clients to a major or a patch version (for example from 24.4.0 to 25.4.0, or from 24.4.0 to 24.4.1).
- If automatic updates are enabled and a major update of standalone clients fails, the clients do not start any jobs until they are updated.
- If automatic updates are enabled and a patch update of standalone clients fails, the clients continue to work, and a warning is displayed.
- You can only update embedded clients and sensors to a patch version (for example, from 24.4.0 to 24.4.1 or 24.4.2, but not to 25.4.0). Automatic updates for major versions is not available for embedded clients and sensors.
- If automatic updates are enabled and a patch update of an embedded client fails, the clients and sensors continue to work, and a warning is displayed.

To update sensors and embedded clients to the next version, you must install the latest OpenText SAST version.

Important

OpenText ScanCentral SAST clients and sensors check for updates only if you use the `-url` or `-sscurl` options. The `package` command does not start the update process.

To enable automatic updates of your clients and sensors:

1. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
2. To enable automatic updates, set `client_auto_update` to `true`.
3. Save and close the file.
4. Restart Tomcat server.

The update process (and its resulting success or failure status) is written to the console.

Important

If you have OpenText SAST installed in a location that requires that you have administrator permissions to modify it (for example in `Program Files`), then to update a sensor, you must start it with administrator permissions. Otherwise, the sensor cannot write files to disk. If automatic updates is enabled, major updates on standalone clients must finish successfully before the sensor can start. With automatic updates enabled, patch updates allow sensors and clients to start unless the upgrade fails.

See also

Upgrading OpenText ScanCentral SAST components

Upgrading the Controller

1.7. Submitting scan requests

You can request a scan that performs remote translation and scan or one that performs a remote scan for a project that is already translated to your OpenText ScanCentral SAST sensors. This content describes how to submit scan requests (including special considerations for some project languages), how to upload your scan results to Application Security in your scan request, and how to prepare a OpenText ScanCentral SAST package to be scanned without sending it to a Controller.

This section contains the following topics:

1.7.1. Submitting local translation and remote scan requests

You can submit a project that OpenText SAST has already translated to your OpenText ScanCentral SAST sensors for remote scanning. To submit a scan request to perform only the scan phase, use the `start` command with either the `--build_id` (`-b`) or the `-mbs` option to identify the local translation or an existing mobile build session file together with the `-scan` option. The following is an example of a scan request to submit a remote scan:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -b <build_id> -scan
```

You can include supported OpenText SAST scan options in the scan request (see Options Accepted for -sargs (--scan-args)). You must include only the OpenText SAST analysis options after the `–scan` option. If the parameter for the option you specify includes a space, you must enclose it in quotes. For example:

```
-scan –build-label "Application 5.4 June 8, yyyy"
```

Note

You can also perform a local translation and remote scan using the OpenText ScanCentral SAST`package` command and the PackageScanner tool. For more details and an example, see Using the PackageScanner tool.

If the scan request is successful, you will receive a job token. The OpenText ScanCentral SAST sensor pulls the scan request from the Controller, processes it, and publishes the results to the Controller.

By default, jobs submitted and scan results (FPR files) cannot be larger than 1 GB. Before you start large scans, review Optimizing Scan Performance.

See also

Performing Remote Translation and Scan Requests

Global Options

Start Command Options

1.7.2. Submitting remote translation and scan requests

If you use a supported language, you can submit your project to your OpenText ScanCentral SAST sensors for a complete remote analysis (both translation and scan phases). To submit a scan request that performs both the translation and scan phases, use the `start` command. For more information, see the *Application Security Software System Requirements* document.

OpenText ScanCentral SAST automatically detects the build tool you are using based on the project files being scanned. For example, if OpenText ScanCentral SAST detects a `pom.xml` file, it automatically sets `-bt` to `mvn`. If it detects a `build.gradle` file, it sets `-bt` to `gradle`. If OpenText ScanCentral SAST detects a `*.sln` file, it sets `-bt` to `msbuild` (Windows) or to `dotnet` (Linux) and sets `-bf` to the `xxx.sln` file. If OpenText ScanCentral SAST detects multiple file types (for example, `pom.xml` and `build.gradle`), it prioritizes the build tool selection as follows: Maven > Gradle > MSBuild and prints a message to indicate which build tool was selected based on the multiple file types found. For a list of supported build tools, see the *Application Security Software System Requirements* document.

The following table provides example scan request commands for different tasks. The examples assume that the command is run from the project's working directory. The build tool option `--build-tool` (`-bt`) shown in these example commands is not required.

Task	**Example command**
Start a job to scan a .NET application.	`scancentral -sscurl<ssc_url> -ssctoken<token> start`
Start a job to scan a dotnet project on Windows.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -btdotnet -bf mySolution.sln`

Start a job to scan an Apache Maven™ Software project that includes the test scope.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -btmvn --include-test`
	or
	`scancentral -sscurl<ssc_url> -ssctoken<token> start -t`

Start a job to scan a Maven project with a non-default build file.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -btmvn -bf c:\myproj\myproj-pom.xml`
Start a job to scan a JavaScript/TypeScript project.	`scancentral -sscurl<ssc_url> -ssctoken<token> start`
Start a job to scan a PHP version 8.2 project.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -hv 8.2`
Start a job to scan an ABAP project.	`scancentral -sscurl<ssc_url> -ssctoken<token> start`
Start a job to scan a Java project and exclude test source files.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -exclude "src/test/**/*"`
Start a job to scan only the distribution files for a JavaScript project.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -include "./dist/**/*.*"`
Start a job to scan all the beta files except for JSON files	`scancentral -sscurl<ssc_url> -ssctoken<token> start -include "./beta/*.*" -exclude "./beta/*.json"`
Start a job to scan a Go project with a build tag.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -targs "-gotags release"`

Start a job to scan a Ruby project.	`scancentral -sscurl<ssc_url> -ssctoken<token> start`
Start a job to scan a Gradle project.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -btgradle`
Start a job to scan a Gradle project, get email notifications from the Controller, and upload the results to Application Security.	`scancentral -sscurl<ssc_url> -ssctoken<token> start -email username@domain.com -upload -application "MyProject" -version "1.0"`

OpenText ScanCentral SAST returns a job token that you can use to track the scan.

See also

Offloading Scanning Only

Global Options

Start Command Options

Submitting Scan Requests and Uploading Results to Application Security

1.7.3. Targeting a specific sensor pool for a scan request

To target a specific sensor pool for a scan request, you must have:

- The name or the UUID for the sensor pool
- The `pool_mapping_mode` property set to enabled or disabled

To get the sensor pool name or the UUID for the sensor pool:

1. Sign in to Application Security and select **ScanCentral**.
2. On the navigation pane of the **SAST** page, select **Sensor Pools**.
3. In the **Sensor Pools** table, copy the value shown in the **Name** or **UUID** column for the sensor pool you want to target for a scan request.

> **Note**
>
> All unassigned and enabled sensors are used, even if they are not assigned to sensor pools.

To specify a sensor pool to use for a scan request:

- At the command prompt on the client host, run one of the following commands:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -pool <pool_name>
```

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -pool <uuid>
```

1.7.4. Requesting job status email notifications for scan requests

To request delivery of email notifications with the status of a scan request, include the `-email` option in the scan request. This is available for both local translation and remote scan and remote translation and scan requests. The following is an example of a remote translation and scan request that will send job status notifications to two email addresses:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -email userA@example.com:userB@example.com
```

By default, the subject in the email notification includes the following information:

```
ScanCentral SAST job <job_status>: <job_token>
```

If the scan request includes an upload of scan results to Application Security, then the subject also includes the application name and application version:

```
ScanCentral SAST job <job_status>: <job_token> (<app_name> - <app_version>)
```

To control whether the job status is included in the email subject, use the `include_job_status_in_email_subject` property in the Controller configuration.

See also

Configuring the Controller

Start command

Viewing the scan request status

1.7.5. Scanning .NET projects

OpenText ScanCentral SASTMSBuild integration is available on Windows only. OpenText ScanCentral SASTdotnet integration is available on Windows and Linux.

To translate and scan .NET projects, the client machine must have the software required to build and package .NET projects installed:

- MSBuild or dotnet

  For supported versions of MSBuild and dotnet, see the *Application Security Software System Requirements* document.

- NuGet (optional)
- .NET Framework, .NET Core, or .NET Standard as required for the project configuration

To use OpenText ScanCentral SASTMSBuild integration, the required MSBuild version must be included in the PATH environment variable. To make sure the project is built correctly, OpenText recommends that you start OpenText ScanCentral SAST from the Developer Command Prompt for Visual Studio, which sets the required .NET environment variables automatically. To use OpenText ScanCentral SASTdotnet integration, the required dotnet version must be included in the PATH environment variable.

Some projects also require that you start NuGet to restore some dependencies. If any dependencies are unresolved, the build fails and the scan results might be incomplete. For these types of projects, you must install NuGet manually on the machine and make sure it is included in the PATH environment variable. If NuGet is found, OpenText ScanCentral SAST runs it automatically.

The following are command-line examples to translate and scan a .NET project:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start --build-tool msbuild --build-file <sln_file_or_path_to_sln_file>
```

```
scancentral -sscurl <ssc_url> -ssctoken <token> start --build-tool dotnet
```

The following command uses MSBuild integration on a Windows client and dotnet integration on a Linux client because no build tool option is specified:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start --build-file <sln_file_or_path_to_sln_file>
```

Note

To use the dotnet integration on a Windows client, you must include `-bt dotnet`.

If no build tool is specified, OpenText ScanCentral SAST client tries to automatically detect the build tool for `*.sln`, `*.csproj`, `*.vbproj`, and `dirs.proj`.

OpenText ScanCentral SAST returns a job token that you can use to track the scan.

Excluding .NET Projects from analysis

To exclude a .NET project from OpenText ScanCentral SAST analysis, you must create a build configuration to exclude the project, and then specify the build configuration with the `--build-command` option.

For example, the solution `MySolution.sln` includes two projects: ProjectA and ProjectB. The `<build_config>` file, created in Visual Studio excludes ProjectB from the builds. To exclude ProjectB from OpenText ScanCentral SAST analysis, run the following from the directory where the solution file resides:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start --build-tool msbuild --build-file MySolution.sln --build-command "/t:Rebuild /p:Configuration=<build_config>"
```

See also

Configuring sensors for remote translation of .NET languages

1.7.6. Scanning older version Java projects

Set the `SCANCENTRAL_JAVA_HOME` environment variable to a version of Java that OpenText ScanCentral SAST supports. For the supported Java versions, see the *Application Security Software System Requirements* document.

Note

This is only required if the JAVA_HOME environment variable is set to Java version that OpenText ScanCentral SAST client does not support.

1.7.7. Scanning JavaScript and TypeScript code

By default, any NPM dependencies (node_modules directory) that exists in your project is included in the project package only for translation. This improves the analysis results by including type resolution information from the JavaScript and TypeScript code. However, OpenText SAST excludes the files in node_modules from the analysis and no vulnerabilities are reported for these NPM dependencies in the scan results.

To prevent OpenText ScanCentral SAST from automatically restoring dependencies, include the `-skipBuild` option in the scan request command. Note that any existing node_modules directory is still included in the project package that is sent to the Controller unless you explicitly exclude it.

To exclude the node_modules directory from your project package, use the `-exclude` option in the `start` or `package` command. For example:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -exclude node_modules
```

To include the NPM dependencies in the scan results, see the *OpenText™ Static Application Security Testing User Guide* for information about translating JavaScript and TypeScript code.

1.7.8. Scanning Python projects

OpenText ScanCentral SAST clients can work with Python® projects in three ways:

- Submit a scan request in a prepared virtual environment (see Starting OpenText ScanCentral SAST in a Virtual Environment).
- Use an existing virtual environment, without activating that virtual environment (see Starting OpenText ScanCentral SAST in an Unactivated Virtual Environment). In this case, OpenText ScanCentral SAST activates the virtual environment.
- Start the job outside of a virtual environment (see Starting OpenText ScanCentral SAST Outside of a Virtual Environment).

The following table provides examples of different ways to submit scan requests for Python code.

Task	**Example command**
Start a job to scan a Python 3 project	`scancentral -sscurl <ssc_url> -ssctoken <token> start --python-requirements <requirements_file_path>`
Start a job to scan a Python 2 project	`scancentral -sscurl <ssc_url> -ssctoken <token> start —-python-version 2 --python-requirements <requirements_file_path>`
Start a job to scan a Python project under an active virtual environment with dependencies already installed	`scancentral -sscurl <ssc_url> -ssctoken <token> start`
Start a job to scan a Python project under an active virtual environment without project dependencies installed	`scancentral -sscurl <ssc_url> -ssctoken <token> start --python-requirements <requirements_file_path>`
Start a job to scan a Python project using an existing Python virtual environment and install project dependencies	`scancentral -sscurl <ssc_url> -ssctoken <token> start --python-virtual-env <venv_location>`

Submitting a scan request in a virtual environment

If you work in a virtual environment, all your project dependencies are already installed. You do not need to invoke the pip package manager before you start the job. OpenText ScanCentral SAST can detect the Python version automatically.

To start the scan job in a virtual environment:

1. At the command prompt, activate the virtual environment.
2. Start a job to scan the Python project as shown in the following example:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start
```

If pip dependencies are not yet installed in the virtual environment used, OpenText ScanCentral SAST installs them automatically using the requirements file with the following example:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start --python-requirements <requirements_file_path>
```

Submitting a scan request in an unactivated virtual environment

To start the scan job in a virtual environment (with all dependencies installed) without activating that virtual environment:

- At the command prompt, start the Python project scan as shown in the following examples:

  ```
  scancentral -sscurl <ssc_url> -ssctoken <token> start --python-virtual-env <venv_location>
  ```

  or

  ```
  scancentral -sscurl <ssc_url> -ssctoken <token> start --python-virtual-env <venv_location> --python-req
  uirements <requirements_file_path>
  ```

OpenText ScanCentral SAST goes to the virtual environment, determines the Python version used, packages all required libraries, and then submits the scan job to the Controller.

Submitting a scan request outside of a virtual environment

To start the scan job if there is no virtual environment on the client, you must have Python installed on the client. If the Python version is not specified in the command, then OpenText ScanCentral SAST uses the first working version from PATH environment variable. OpenText ScanCentral SAST locates the Python installation. In this case, OpenText ScanCentral SAST creates a temporary virtual environment, installs all dependencies from the requirements file, and then submits the job to the Controller.

To start the scan job outside of a virtual environment:

- At the command prompt, start the scan job as shown in the following example:

  ```
  scancentral -sscurl <ssc_url> -ssctoken <token> start --python-version 3
  ```

To prevent OpenText ScanCentral SAST from automatically restoring dependencies using `pip install`, include the `-skipBuild` option in the scan request command. If dependencies were already restored before running OpenText ScanCentral SAST, they are included in the project package that is sent to the Controller.

1.7.9. Scanning Go projects

To enable OpenText ScanCentral SAST clients to package Go projects for remote translation and scan, the following requirements must be met:

- The Go compiler must be installed on the client to resolve project dependencies.
- The Go compiler executable location must be available in the PATH variable.
- Configure the Go environment variables. For example, to use a specific Go proxy, configure it as follows:

  `set GOPROXY=`.... (Windows)

  `export GOPROXY=`... (Linux)

  Note

  Sensors do not require a connection to a Go proxy website to resolve dependencies because they run Go translation with `GOPROXY=off` configured. Also, the vendor directory under the project root has all the required dependencies. The sensor rewrites the `GOFLAGS` system variable with `GOFLAGS=-mod=vendor` when it runs an OpenText SAST translation.

- The Go project must include a `go.mod` file.
- OpenText recommends that the Go project includes a `go.sum` file to ensure that dependencies restored with `go mod vendor` works successfully.

To prevent OpenText ScanCentral SAST from automatically restoring dependencies using `go mod vendor`, include the `-skipBuild` option in the scan request command. If dependencies were already restored before running OpenText ScanCentral SAST, they are included in the project package that is sent to the Controller.

1.7.10. Scanning PHP projects

If your PHP project uses the Composer dependency manager and you want to include dependencies in the analysis, then do the following on the client machine:

- Install PHP and Composer
- Configure the `php.ini` to run Composer for your project

This enables OpenText ScanCentral SAST client to invoke Composer to restore the dependencies before packaging the project for analysis. To prevent OpenText ScanCentral SAST from automatically restoring dependencies, include the `-skipBuild` option in the scan request command. If Composer already restored the dependencies before running OpenText ScanCentral SAST, they are included in the project package that is sent to the Controller. If Composer is not configured for your project, then OpenText ScanCentral SAST packages the project without restoring the dependencies.

1.7.11. Scanning COBOL projects

OpenText ScanCentral SAST clients can package COBOL projects for remote translation and scan. For detailed information about the requirements and options available for COBOL analysis, see the *OpenText™ Static Application Security Testing User Guide*.

You must have a sensor with the Windows operating system. OpenText ScanCentral SAST automatically assigns COBOL scans to a Windows sensor. If no Windows sensor is available, then the scan job is created but cannot be started.

Make sure the copybook files are in a separate directory from the COBOL source code files. OpenText recommends that you place your COBOL source code files in a directory called sources and your copybook files in a directory called copybooks. Create these directories at the same level.

Note

To analyze a COBOL project on Linux and to use Legacy COBOL translation, you must perform a local OpenText SAST translation:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -b <build_id>
```

The following example command submits a scan request for a COBOL project where the copybooks files are in the local `copybooks` directory:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -targs "-copydirs copybooks -dialect COBOL390"
```

The following example command submits a scan request for a COBOL project that contains source code files with a non-standard file extension `mfcbl`:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -targs "-copydirs MyCopydir1;MyCopydir2 -Dcom.fortify.sca.fileextensions.mfcbl=COBOL"
```

The following example command submits a scan request for a COBOL project that contains source code files without file extensions:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -targs "-copydirs MyCopyDir -noextension-type COBOL"
```

1.7.12. Scanning SQL projects

On Windows (and Linux for .NET projects only), OpenText SAST assumes that files with the `.sql` extension are T-SQL rather than PL/SQL. To perform a remote translation of a SQL project, you might need to specify what type of SQL your project uses.

To scan the project, run one of the following commands:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -targs "-sql-language PL/SQL"
```

or

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -targs "-sql-language TSQL"
```

1.7.13. Uploading results to OpenText Application Security

To submit a scan request and upload the scan results to an application version in Application Security, you must have an authentication token of type ScanCentralCtrlToken. You can create an authentication token with the fortifyclient utility or in Application Security. You can reuse the token for future requests. The fortifyclient utility is provided with Application Security and the OpenText Application Security Tools installation. For more information about creating authentication tokens with the fortifyclient utility or in Application Security, see the *OpenText™ Application Security User Guide*.

There are two options for providing upload permission, which depend on the permissions you want to give to your Application Security users:

- The user assigned a role that has **Run ScanCentralSAST scans**, **View ScanCentralSAST**, **View application versions**, and **Upload analysis results** permissions generates the token.
- The user assigned a role that has the **Run ScanCentralSAST scans** and **View ScanCentralSAST** permissions (and does not have the **Upload analysis results** permission) generates the token and the Controller is configured with a OpenText ScanCentral SAST Controller service account.

  Use this option to upload the scan results to Application Security using the Controller service account.

  To configure a OpenText ScanCentral SAST Controller service account:

  1. In Application Security, create a OpenText ScanCentral SASTController service account that has the **ScanCentralSAST Controller** role.

     For instructions on how to create Application Security user accounts, see the *OpenText™ Application Security User Guide*.

  2. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
  3. Specify the credentials for the OpenText ScanCentral SAST Controller service account in the `ssc_ctrl_account_username` and `ssc_ctrl_account_password` properties.
  4. Save and close the `config.properties` file.
  5. To apply the change, restart the Controller.

Note

The **Run ScanCentralSAST scans** permission and the **ScanCentralSAST Controller** role are available in Application Security version 24.4.0 and later. To use an earlier version of Application Security, you must do one of the following:

- Ensure that the account of the user that generates the token has a role that includes the **Upload analysis results** and **View ScanCentralSAST** permissions.
- Configure the Controller (steps b-e in the previous procedure) with a OpenText ScanCentral SASTController service account created in Application Security that has a role that includes the **View ScanCentralSAST**, **View application versions**, and **Upload analysis results** permissions.

Examples of scan requests that upload scan results

The following example scan requests perform a remote translation and scan and upload the scan results:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -upload -versionid <app_version_id>
```

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -upload -application <app_name> -version <app_version>
```

The following example scan request performs a local translation and remote scan and uploads the scan results:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -upload -versionid <app_version_id> -b <build_id> -scan
```

See also

Retrying Failed Uploads to Application Security

Global Options

Start Command Options

Submitting Remote Translation and Scan Requests

Submitting Remote Scan Only Requests

1.7.13.1. Specifying a scan results (FPR) file name

You can specify the name of the scan results (FPR) file you upload to Application Security using the `-fprssc` option with the `start` command. The file name must not exceed 128 characters in length and *must not* contain the following characters:

- colon (:)
- backslash (\)
- forward slash (/)
- asterisk (*)
- question mark (?)
- vertical bar or pipe (|)
- less than (<)
- greater than (>)
- double quote (")

The following example scan request performs a remote translation and scan and specifies a name for the FPR file to upload:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -upload -versionid <app_version_id> -fprssc <my_fpr>.fpr
```

The following example scan request performs a remote scan and specifies a name for the FPR file for upload:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -upload -versionid <app_version_id> -fprssc <my_fpr>.fpr -b <build_id> -scan
```

See also

Global Options

Start Command Options

1.7.13.2. Preventing replacement of duplicate scan requests

A duplicate scan request occurs if you have more than one scan request that uploads scan results to the same application version in Application Security. By default, the Controller is configured to replace duplicate scan jobs (`replace_duplicate_scans` property). You can prevent the replacement for specific scan requests with the `--disallow-replacement` (`-dr`) option in a scan request.

Consider the following scenario:

1. Submit a scan for upload to AppA 1.0, scan job 1 is added to the queue.
2. Submit a scan for upload to AppA 1.0, scan job 1 is canceled and scan job 2 is added.
3. Submit a scan for upload to AppA 1.0 with the `-dr` option, scan job 2 is canceled and scan job 3 is added to the queue.
4. Submit a scan for upload to AppA 1.0 with or without the `-dr` option, scan job 3 remains in the queue and scan job 4 is added to the queue.

The following example scan request performs a remote translation and scan, uploads the results to the application version AppA, 1.0 on Application Security, and overrides a duplicate replacement to ensure the scan job is not removed from the queue by future scan requests uploaded to the same application version:

```
scancentral -sscurl <ssc_url> -ssctoken <token> start -upload -application AppA -version 1.0 --disallow-replacement
```

See also

Configuring the Controller

1.7.13.3. Retrying failed uploads to OpenText Application Security

If a job configured to upload scan results to Application Security fails, the Controller retries to upload (up to five attempts by default) and, if the next attempt fails, waits two minutes before it tries again.

If the Controller fails to upload an FPR file, you can use the upload command as follows to resend the FPR:

```
scancentral -sscurl <ssc_url> -ssctoken <token> upload -token <job_token>
```

where *<job_token>* is the token for original job that failed to upload the FPR.

See also

Configuring FPR Upload Retry Attempts

Submitting Scan Requests and Uploading Results to Application Security

1.7.13.4. Configuring upload to OpenText Application Security retry attempts

To configure the number of times the Controller can retry to upload scan results, and the amount of time the Controller waits after a failed upload before it tries again:

1. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/config.properties` file in a text editor.
2. To set the maximum number of upload retry attempts, locate the `ssc_upload_retry_count` property, and replace the default value of `5` with any integer value from `1` to `10`.

   Note

   If the specified value is outside of the valid range or is invalid, OpenText ScanCentral SAST applies the default value.

3. To set the interval between upload retry attempts, locate the `ssc_upload_retry_interval` property, and replace the default value of `120` (seconds) with any integer value from `60` (1 minute) to `900` (15 minutes).

   Note

   If the specified value is outside of the valid range or is invalid, OpenText ScanCentral SAST applies the default value.

4. Save and close the `config.properties` file.

See also

Submitting Scan Requests and Uploading Results to Application Security

Retrying Failed Uploads to Application Security

1.7.14. Optimizing scan performance

If you plan to regularly scan large applications, OpenText recommends that you run a manual test scan on hardware that is equivalent to the hardware on which your sensor is installed.

To optimize your scan:

1. Set the OpenText SAST scan parameters for optimal performance by adjusting the memory settings to align with your hardware.

   For information about how to tune OpenText SAST, see the *OpenText™ Static Application Security Testing User Guide*.

2. Run a scan.
3. Note the size of the resulting FPR file and scan log.
4. To ensure that the Controller and Application Security can accept FPR or log files larger than 1 GB, increase the maximum upload size threshold by doing the following:

   1. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/classes/config.properties` file.

   2. Set the Controller threshold to the maximum size in megabytes as follows:

      ```
      max_upload_size=<max_size_in_megabytes>
      ```

      The default value is 1024.

5. Make sure that OpenText SAST is configured to process large FPR files.

   For more information, see the *OpenText™ Static Application Security Testing User Guide*.

See also

Configuring the Controller

1.7.15. Generating an OpenText ScanCentral SAST package

Use the package command to create a ZIP archive for the specified project. The ZIP archive project package includes the following information:

- Libs—Folder that contains the project dependencies (Gradle, Maven, MSBuild, Java, and .NET projects)
- Src—Folder that contains the source files
- metadata—Specification file that the sensor uses to generate OpenText SAST commands

The following table provides examples of different commands to generate a project package with OpenText ScanCentral SAST client. The examples assume that the command is run from the project's working directory. In these examples, OpenText ScanCentral SAST client creates a package with the name `fortifypackage.zip` unless the `-o` option is used to specify a custom package name.

Note

OpenText ScanCentral SAST client can automatically detect the build tool you are using based on the project files being scanned so use of the `--build-tool` (`-bt`) option is usually not required.

Task	Example command
Create a package from a dotnet project on Linux.	`scancentral package`
Create a package from an MSBuild project.	
Create a package from a dotnet project on Windows.	`scancentral package -bt dotnet`
Create a package from a Gradle project.	`scancentral package`
Create a package from a Maven project with a custom pom.xml file.	`scancentral package -bfmyCustomPom.xml`
Create a package from an ABAP project.	`scancentral package`
Create a package from an Apex project.	`scancentral package`
Create a package from a Classic ASP project.	`scancentral package`
Create a package from a COBOL project.	`scancentral package -targs "-copydirs copybooks" -targs "-dialect COBOL390"`
Create a package from a ColdFusion (CFML) project.	`scancentral package`
Create a package from a Java project.	`scancentral package`
Create a package with the name MyPackage.zip from a Java project.	`scancentral package -o MyPackage.zip`
(For use with OpenText™ Core Application Security only) Create a package from a Java project and include additional files required for open source software composition analysis.	`scancentral package -oss`
Create a package from a Java project and exclude test source files.	`scancentral package -exclude "./src/test/**/*"`

Create a package from a JavaScript/TypeScript project that only includes the distribution files.	`scancentral package -include "./dist/**/*.*"`
Create a package of all the beta files except for JSON files	`scancentral package -include "./beta/*.*" -exclude "./beta/*.json" -o BetaWithoutJSON.zip`
Generate a package from an Android project in Kotlin that uses the Android plugin.	`scancentral package -bt gradle`
Create a package from a Go project.	`scancentral package`
Create a package for only IaC/Dockerfiles. **Note** If Dockerfiles are included in a Gradle, Maven, or MSBuild project, then the Docker files are automatically included in the package.	`scancentral package`
Create a package from a PHP project.	`scancentral package`
Create a package from a Python 2 project.	`scancentral package -yv 2 -pyr<requirements_file_path>`
Create a package from a Python project under an active virtual environment with dependencies already installed.	`scancentral package`
Create a package from a Python project under an active virtual environment without project dependencies installed.	`scancentral package -pyr<requirements_file_path>`
Create a package from a Python project using an existing Python virtual environment and install project dependencies.	`scancentral package -pyv<venv_location>`
Create a package from a Ruby project.	`scancentral package`
Create a package from a SQL project.	`scancentral package -targs "-sql-language TSQL"` `scancentral package -targs "-sql-language PL/SQL"`
Create a package from a Visual Basic project.	`scancentral package`

See also

Package Command Options

Using the PackageScanner Tool

1.7.15.1. Open source software composition analysis (OpenText Core Application Security only)

OpenText Core Application Security (Fortify on Demand) customers can use the `--open-source-scan` (`-oss`) option with the `package` command to include additional files required for open source software composition analysis by OpenText Core SCA. By default, the OpenText ScanCentral SAST client uses the Debricked CLI to automatically generate the lock files required for open source composition analysis. Using the Debricked CLI, gives you the most up-to-date Debricked artifact generation. OpenText ScanCentral SAST client installs the Debricked CLI if it is not yet installed and checks for a newer version online.

The OpenText ScanCentral SAST client installs the Debricked CLI in one of the following locations:

- Default location:
  - On a Windows system: `%LOCALAPPDATA%\Fortify\scancentral-<version>\debricked\`
  - On a Linux system: `<userhome>/.fortify/scancentral-<version>/debricked/`
- Custom location specified by the `debricked_cli_dir` property in the `<client_install_dir>/Core/config/client.properties` file

If you want to use the Debricked CLI without the automatic installation, you can manually place the Debricked CLI in either location. See the Debricked CLI documentation for instructions on how to download the latest releases. To avoid automatic updates of the Debricked CLI, include the `--skip-debricked-update` (`-sdu`) option in your OpenText ScanCentral SAST client package command.

If you want to prepare the files by yourself using the Debricked CLI directly and you don't want the OpenText ScanCentral SAST client to overwrite the prepared files, use the `--debricked-no-resolve` or `-dnr` option in your OpenText ScanCentral SAST client package command.

1.7.16. Using the PackageScanner tool

If you have OpenText SAST locally installed, you can run an analysis of a project package without sending it to the Controller. The PackageScanner tool takes a project package created by the OpenText ScanCentral SAST client `package` command, generates OpenText SAST commands, and then translates and scans it using a locally installed OpenText SAST.

You can also use the PackageScanner tool to perform only a translation on the project package and then submit the project package to the Controller for analysis.

You can find the PackageScanner tool in the `<sast_install_dir>/bin/` directory. To configure the location of PackageScanner log, use the SCANNER_LOG environment variable. For example, `export SCANNER_LOG=/mylogs/packagescanner`.

The default value for PackageScanner:

- Windows: `<user_app_data>\Local\Fortify\packagescanner-<version>\log`
- Linux: `<user_home>\.fortify\packagescanner-<version>\log`

Note

You can set Java system properties for the PackageScanner tool to use by adding them to the SCANCENTRAL_VM_OPTS environment variable. For example, to specify a temp directory that has a short path in Windows, type:

```
set SCANCENTRAL_VM_OPTS=-Djava.io.tmpdir=C:\mytemp
```

The following table describes the PackageScanner tool command-line options.

Packagescanner option	Description
`-f`, `--fpr <file>.fpr`	(Optional) Specifies the FPR file to which scan results are written. This option required unless you are including the `--no-scan` option to perform the translation only on the project package.
`-p`, `--package <package_name>.zip`	(Required) Specifies the path to the project package file generated by the OpenText ScanCentral SAST client with the `package` command.
`-b`, `--build-id <id>`	(Optional) Specifies the build ID. OpenText SAST uses the build ID to track which files are compiled and combined as part of a build, and later, to scan those files. If you do not specify a build ID, PackageScanner automatically generates one.  **Important** This option is required if you are including the `--no-scan` option so that you can perform the scan later with OpenText SAST after the translation phase is complete.
`-noscan`, `--no-scan`	(Required to skip the scan phase) Specifies for PackageScanner to only translate the project package and no scan phase is performed. Include the build ID (`--build-id` option) with this option. Use this option if you plan to perform the scan phase as a separate step.

`-sca`, `--sca-path <path>`	(Optional if started from OpenText SAST) Specifies the path to the OpenText SAST executable. If the OpenText ScanCentral SAST client is part of the OpenText SAST installation (embedded), the path is determined automatically.
`-targs`, `--translation-arguments<translation_options>`	(Optional) Specifies OpenText SAST translation options. Enclose multiple options in quotes separated by spaces or repeat this option for each OpenText SAST option and parameter.
`-sargs`, `--scan-arguments <scan_options>`	(Optional) Specifies OpenText SAST scan options. Enclose multiple options in quotes separated by spaces or repeat this option for each OpenText SAST option and parameter.
`-tlog`, `--sca-translation-log <log_file_path>`	(Optional) Specifies a log file for translation commands. By default, PackageScanner creates the log file in a temporary directory, which is removed after the program execution.
`-slog`, `--sca-scan-log <path>`	(Optional) Specifies a log file for scan commands. By default, PackageScanner creates the log file in a temporary directory, which is removed after the program execution.
`-workdir`, `--working-dir <dir>`	(Optional) Specifies a directory where the project package is unpacked and PackageScanner creates the OpenText SAST project root directory. By default, PackageScanner creates this directory in a temporary location and removes it after the program execution (unless the `-debug` option is specified).
`-debug`	(Optional) Enables debug logging for OpenText ScanCentral SAST clients and sensors.
`-v`, `--version`	(Optional) Displays the PackageScanner tool version.

The following are example PackageScanner commands:

```
packagescanner --package package.zip --fpr results.fpr
packagescanner --package package.zip --fpr results.fpr --translation-arguments "-debug -verbose" --scan-argu
ments "-debug -verbose"
packagescanner --package JavaApackage.zip --fpr results.fpr --translation-arguments "-build-label myJavaBuild
A"
packagescanner --package package.zip --fpr results.fpr --sca-translation-log trans.log --sca-scan-log scan.log
packagescanner --package package.zip --fpr results.fpr --sca-path C:\appsecurity\bin\sourceanalyzer.exe
packagescanner --package package.zip --fpr results.fpr --working-dir C:\packageScannerTemp
```

The following example performs a local translation and a remote scan by creating a project package with the package command, performing the local translation with the PackageScanner tool, and then submitting the scan to the Controller:

```
scancentral package -o MyProjPackage.zip
packagescanner -package MyProjPackage.zip -b xyz --no-scan
scancentral -sscurl <ssc_url> -ssctoken <token> start -b xyz -scan
```

See also

Generating a ScanCentral SAST Package

1.8. Managing scan requests and scan results

This section describes how to view the status of your scan requests, retrieve the scan results, and cancel scan requests from the OpenText ScanCentral SAST client command line.

You can also manage scan requests and obtain scan results from Application Security. For more information, see *OpenText™ Application Security User Guide*.

This section contains the following topics:

1.8.1. Viewing the scan request status

To view the status of a OpenText ScanCentral SAST scan request, run the following command:

```
scancentral –url <controller_url> status -token <job_token>
```

You can also view the scan request status from Application Security. For instructions, see the *OpenText™ Application Security User Guide*.

The following table lists the possible values for OpenText ScanCentral SAST scan request and upload status, which are available in the console, the scan logs, and in Application Security. The `SSC upload status` is provided only for scan requests that include uploading the scan results (FPR file) to Application Security.

Status type	Status	Description
Job status	PENDING	The Controller accepted the scan job.
	QUEUED	Scan job was assigned to a sensor.
	CANCELED	Scan was canceled.
	RUNNING	Scan is currently running.
	FAILED	Scan failed due to an OpenText SAST error.
	FAULTED	Scan failed due to a sensor error.
	TIMEOUT	Scan was canceled due to timeout.
	COMPLETED	Scan completed successfully.
SSC upload status	PENDING	Request to upload the scan results (FPR) is pending.
	QUEUED	A scan results (FPR) upload is awaiting upload to Application Security.
	CANCELED	A scan results (FPR) upload to Application Security was canceled or failed.
	FAILED	A scan results (FPR) upload to Application Security failed.
	COMPLETED	A scan results (FPR) file was uploaded to Application Security successfully.

See also

Status Command Options

1.8.2. Retrieving scan results from the Controller

To retrieve scan results, run the following command:

```
scancentral -url <controller_url> retrieve -token <job_token> -f <results>.fpr -log <my_log>.log
```

See also

Retrieve Command Options

1.8.3. Canceling scan requests

To cancel a scan request, run the following command:

```
scancentral –url <controller_url> cancel -token <job_token>
```

You can also cancel scan requests in Application Security from the **ScanCentral** view. For instructions, see the *OpenText™ Application Security User Guide*.

See also

Cancel Command Options

1.9. Troubleshooting

The following topics provide information on how to troubleshoot problems you might encounter working with OpenText ScanCentral SAST and how to gather information for Customer Support.

This section contains the following topics:

1.9.1. Locating log files

The following table describes where to find the log files for different components.

Component	Operating system	Default Log file location
Controller	Windows Linux	`<controller_install_dir>/tomcat/logs/scancentralCtrl.log` For information about changing the log file location, see Configuring the Controller logging. For information about how to configure the logging level for the Controller, see Configuring the Logging Level on the Controller.
Sensor Client	Windows	`%LOCALAPPDATA%\Fortify\scancentral-<version>\log\`
	Linux	`<userhome>/.fortify/scancentral-<version>/log/`
PackageScanner	Windows	`%LOCALAPPDATA%\Fortify\packagescanner-<version>\log\`
	Linux	`<userhome>/.fortify/packagescanner-<version>/log/`

1.9.2. Troubleshooting the Controller

After upgrading the binaries on the local server for the Controller, you can access the Controller using the address `<protocol>://<controller_host>:<port>/scancentral-ctrl/`, but you cannot access it from the workstation. Also, while trying to integrate Application Security with the Controller, the Controller status is not visible, even though the `config.properties` file was updated with the required details.

Open the `<client_install_dir>/Core/Config/client.properties` file to make sure that the value set for the `client_auth_token` property matches the value for the same property in the `config.properties` file found in your Controller installation directory.

1.9.3. Troubleshooting a sensor as a Windows service

To troubleshoot issues encountered during the configuration of sensor auto-start as a Windows service, review the logs listed in the following table.

Log type	**Default log file location**[1]
Primary OpenText ScanCentral SAST sensor log	`C:\Windows\System32\config\systemprofile\AppData\Local\Fortify\scancentral-<version>\log\scancentral.log`
Sensor temporary directories that contain MBS files, OpenText SAST log files, and generated FPR files	`C:\Users\Public\Fortify\SC\<job_token>`
Sensor `stdout` and `stderr` logs **Note** Before you start a sensor, check to make sure that the log files are not open in an application. Open log files prevent procrun from writing to the file.	`C:\Users\Public\Fortify\SC\workerout.log` `C:\Users\Public\Fortify\SC\workererr.log`
Commons-daemon log	`C:\Users\Public\Fortify\SC\<year_month_day>.log`

[1] The log file location might be different if you changed the account under which the service is run, or you have set the WORKDIR environment variable.

If you experience an issue starting a OpenText ScanCentral SAST sensor that you installed as a Windows service and the log files do not include enough information to resolve the issue, you can run the service as a console application to get more information. Run the following commands from an administrator command prompt:

```
cd <sast_install_dir>\bin\scancentral-worker-service
prunsrv.exe //TS//FortifyScancentralWorkerService
```

This enables you to see any service startup errors that might help you to troubleshoot the issue.

1.9.4. Preserving the OpenText SAST project root directory

By default, the OpenText ScanCentral SAST sensor creates a temporary working directory to unpack the project package and store temporary files for the scan including the OpenText SAST project root directory. This working directory is automatically deleted after the scan unless the `-debug` option is provided in the scan request. You can also configure an option to prevent the OpenText SAST project root directory from being deleted. To preserve the OpenText SAST project root directory:

1. Open the `<sast_install_dir>/Core/config/worker.properties` file in a text editor.
2. Look for the `delete_sca_build_dir` property and set it to `false`.
3. Save the changes.

After the scan is complete, you can find the OpenText SAST project root directory in the job directory, which is in one of the following locations:

- The `jobs` directory in the sensor's working directory
- In the directory configured with the `jobs_dir` property in the `worker.properties` file

See also

Configuring Where to Generate Job Files and the worker_persist.properties File

1.9.5. Configuring the log level on the Controller

OpenText ScanCentral SAST logs typically provide enough information to follow the flow of operations under normal conditions. If things are not working as expected, the logging might not provide enough information to determine the actual root cause of the issue. If the Controller log information is insufficient, you can increase the amount of information by changing the log level. The following instructions describe how to configure the log level on the Controller. For instructions on how to change the log level on sensors and clients, see Enabling debugging on clients and sensors.

To configure the log level on the Controller:

1. Open the `<controller_install_dir>/tomcat/webapps/scancentral-ctrl/WEB-INF/classes/log4j2.xml` file in a text editor.
2. Locate one of the following strings:
   - `<Logger name="com.fortify.cloud" level="info" additivity="false">`
   - `<Logger name="com.fortify.cloud.ctrl.service" level="info" additivity="false">`
3. For a more detailed level of logging, change the level as shown in the following example:

   ```
   <Logger name="com.fortify.cloud" level="debug" additivity="false">
   ```

4. To apply the change, restart the Controller.

For more information about log levels and defining custom log levels, see the Apache Logging Services website

See also

Enabling debugging on clients and sensors

Configuring the Controller logging

Locating log files

1.9.6. Enabling debugging on clients and sensors

The client and sensor logs typically provide enough information to follow the flow of operations under normal conditions. If things are not working as expected, the logging might not provide enough information to determine the actual root cause of the issue. If the client or sensor log information is insufficient, you can increase the log level by adding the `-debug` command-line option to the OpenText ScanCentral SAST command. Always specify the `-debug` option *before* you specify the command.

Examples:

```
scancentral -debug -url <controller_url> worker
scancentral -debug -url <controller_url> start
```

The next time the sensor is called, the log contains debug-level information.

See also

Configuring the Log Level on the Controller

Locating Log Files

1.9.7. Creating a log archive for Customer Support

If you are experiencing any issues with OpenText ScanCentral SAST, you can use the `-diag` option for the `start` command to generate a ZIP file that includes debug log files from clients, sensors, and OpenText SAST. You can share this ZIP when you contact Customer Support.

The following is an example command to generate the archive:

```
scancentral -url <controller_url> start --diagnosis <debug_data.zip>
```

The generated ZIP file contains the following:

- Client debug log entries for the specific scan invocation only
- Sensor debug log entries for the specific job
- The Support log from OpenText SAST
- MSBuild or dotnet build log

  Included only when scanning .NET projects.
- Metadata file from the project package

  Included when using remote translation and scan.

1.10. OpenText ScanCentral SAST command-line options

This section describes the command-line options that you can use with OpenText ScanCentral SAST client.

This section contains the following topics:

1.10.1. Global options

This topic describes the global command-line options that you can use with OpenText ScanCentral SAST client. You must specify these options before any of the commands described in the following sections.

Global option	Description
`-h`, `--help` *`<command>`*	Displays help for the selected command. To see all command help, type `-h all`.
`-v`, `--version`	Displays the OpenText ScanCentral SAST version.
`-sscurl` *`<web_address>`*	Specifies the web address of a Application Security server that is integrated with the Controller. You must include the `-ssctoken` option with this option for authentication.
`-ssctoken` *`<token>`*	Specifies a Application Security authentication token of type ScanCentralCtrlToken. For information about how to acquire authentication tokens, see the *OpenText™ Application Security User Guide*. You must include the `-sscurl` option with this option to specify the Application Security server.
`-url` *`<web_address>`*	Specifies a Controller web address. If you upload scan results to Application Security, then the Controller must be integrated with a Application Security instance. **Note** Do not include the `-sscurl` and `-ssctoken` option pair with this option.
`-debug`	Enables debug logging on OpenText ScanCentral SAST clients and sensors. For information on how to configure the logging level on the Controller, see Configuring the Logging Level on the Controller.

1.10.2. Start command

Use the `start` command to perform a remote scan, or to perform a remote translation and scan.

Start command option	Description
Options for all scan requests	
`-upload`, `--upload-to-ssc`	Uploads the scan results to Application Security after completion. For more information about uploading scan results, see Uploading results to Application Security.
`-application <name>`	Specifies the Application Security application name. The `<name>` value is case-sensitive.
`-version`, `--application-version <name>`	Specifies the Application Security application version name. The `<name>` value is case-sensitive.
`-versionid`, `--application-version-id <id>`	Specifies the Application Security application version ID.
`-uptoken`, `--ssc-upload-token <token>`	Specifies the Application Security authentication token of type ScanCentralCtrlToken, which is only required if you are uploading scan results and specify the Controller with the global `-url` option. **Note** If the `pool_mapping_mode` property is set to `disabled`, you can also use a token of type AnalysisUploadToken. For information about how to acquire authentication tokens, see the *OpenText™ Application Security User Guide*.
`-fprssc`, `--fpr-filename-on-ssc <file>`	Specifies the name to use for the FPR file uploaded to Application Security. For more information about this option, see Specifying a scan results (FPR) file name.
`-dr`, `--disallow-replacement`	Prevents a scan job from being replaced because it is a duplicate (targeted for upload to the same application version as an existing queued scan job). For more information about this option, see Preventing replacement of duplicate scan requests.
`-block`	Waits for the job to complete, and then downloads the scan results from the Controller.
`-f`, `--output-file <file>`	Specifies the name for the local FPR file output. Use with the `-block` option to specify the name for the local FPR file output after a scan is completed.
`-diag`, `--diagnosis <zip_file>`	Generates a ZIP file that includes debug log information from client, sensor, and OpenText SAST that Customer Support requires to analyze any problems you might encounter. For more information about this option, see Creating Archive Logs for Customer Support.

`-email <address>`	Specifies the address for job status notifications. To send the notification to multiple email addresses, specify a colon-, comma-, or semicolon-separated list of email addresses. You can specify a maximum of 100 email addresses. For example:
	`-email userA@example.com:userB@example.com`
	Use of a colon to separate multiple email addresses works in most shells. If you use shell that interprets colon, comma, or semicolon as a delimiter, then you must enclose multiple email addresses in quotes. For example:
	`-email "userA@example.com;userB@example.com"`

`-filter <file>`	Specifies a filter file to use during a scan (repeatable).
`-log`, `--log-file <file>`	Specifies a file name for the local log file after the scan is complete.
`-slog`, `--sensor-log-file <file>`	Use with the `-block` option to specify the file name for the local sensor log output.
`-j`, `--job-file <file>.zip`	Specifies a file name for the local job file that was submitted to OpenText ScanCentral SAST for analysis. The job file for remote translation contains the project package (sources, dependencies, and metadata). The job file for local translation contains the mobile build session (MBS) file. Use with the `-block` option.
`-o`, `--overwrite`	Overwrites the existing FPR or log with new data.
`-projtl`, `--project-template <file>`	Specifies an issue template file to include.
`-pi`, `--poll-interval <n>`	Specifies how often (in seconds) to poll the processing status. The valid range for *<n>* is from `10` to `60`.
`-pool`, `--submit-to-pool <uuid>` | *<pool_name>*	Specifies a specific sensor pool for the scan request. You can specify the sensor pool by either the UUID or the pool name.
`-sto`, `--scan-timeout <n>`	Specifies the maximum amount of time (in minutes) a sensor can work on an assigned job (and prevent the sensor from doing other work). Use of this option has a higher priority than the `scan_timeout` property setting in the `config.properties` file.
`-rules <file/dir>`	Specifies a custom rules file or directory to use during the scan (repeatable).
`-sp`, `--save-package <file>`	Specifies the project package file to save after submitting the scan request. The *<file>* must have a `*.zip` extension. This project package contains the following information: • Libs—Folder that contains the project dependencies (Gradle, Maven, MSBuild, Java, and .NET projects) • Src—Folder that contains the source files • metadata—Specification file that the sensor uses to generate OpenText SAST commands
Options for local translation and remote scan requests	

`-b`, `--build-id <id>`	Specifies the build ID of a previously translated project to upload to the Controller for analysis.
`-mbs <file>`	Specifies a mobile build session file for a previously translated project to upload to the Controller for analysis.
`-projroot`, `--project-root <dir>`	Specifies the project directory for the mobile build session export.
`-scan`	Sets the point beyond which all options are for OpenText SAST.
Options for remote translation and scan requests	
`-p`, `--package <file>`	Specifies the project package file to upload to the Controller (see Package Command).
`-bt`, `--build-tool <name>`	Specifies the build tool used for the project. The valid values for *<name>* are `dotnet`, `gradle`, `msbuild` (Windows only), `mvn`, or `none`. The following example specifies a maven project with build parameters: `-bt mvn -bc "package --setting custom.xml"` If not specified, OpenText ScanCentral SAST automatically detects the build tool based on the project files being scanned.
`-bc`, `--build-command <commands>`	(For use with Maven, Gradle, dotnet, and MSBuild) Specifies custom build parameters for preparing and building a project. The following example build command starts a Gradle build before packaging the project: `-Prelease=true clean customTask build` If you use the `-bc` option and the build fails, OpenText ScanCentral SAST stops working on the build. (Gradle only) If you *do not* use `-bc`, the default command, default tasks, and target are invoked. If the build fails, OpenText ScanCentral SAST displays a warning, but continues to work and then displays a message to indicate that the build procedure failed and your results might be incomplete.
`-bf`, `--build-file <file>`	Specifies the build file if you are not using a default name such as `build.gradle` or `pom.xml`.
`-q`, `--quiet`	Prevents the printing to stdout from the build execution.
`-skipBuild`	Disables the project preparation build step if your projects uses Gradle or Maven before packaging. If you use this option, any `-bc` option specified is ignored. If your project does not use a build tool, you can use this option to prevent OpenText ScanCentral SAST from automatically restoring dependencies using a package manager (for languages such as Go, JavaScript/TypeScript, PHP, and Python).
`-t`, `--include-test`	Includes the test source set (Gradle), the test scope (Maven), or projects in your solution that reference NUnit, xunit, or MSTest (.NET).
`-exclude <file_paths>`	Specifies files or directories (with absolute or relative path, or Ant-style path pattern) to exclude from the analysis (repeatable). Separate multiple file paths with semicolons (Windows) or colons (Linux). For example, you might use this option to exclude a few test files from the analysis.

`-include <file_paths>`	Specifies files or directories (with absolute or relative path, or Ant-style path pattern) to include in the analysis (repeatable). Only file paths for files within the current working directory are included. Separate multiple file paths with semicolons (Windows) or colons (Linux). For example, you might use this option if you have only a few files you want to include in the analysis. You can combine this option with the `-exclude` option to exclude specific files from the included path. See Submitting remote translation and scan requests for example commands.
`-hv`, `--php-version <version>`	Specifies the PHP version. If not specified, OpenText ScanCentral SAST automatically detects the installed PHP version.
`-pyr`, `--python-requirements <file>`	Specifies the Python project requirements file to install and collect dependencies.
`-pyv`, `--python-virtual-env <dir>`	Specifies the Python virtual environment location.
`-yv`, `--python-version <version>`	Specifies the Python version. The valid values are `2` and `3`. This option is ignored if OpenText ScanCentral SAST client is started under a Python virtual environment or if `--python-virtual-env` is specified.
`-targs`, `--translation-args <translation_option>`	Specifies an OpenText SAST translation option (repeatable). For multiple translation options, use multiple `-targs` options. If the translation option has a path parameter that includes a space, enclose the path in single quotes. For a list of OpenText SAST options you can use with the `-targs` option, see Options accepted for -targs (--translation-args). If you use the `-targs` option with the `--package` option, OpenText ScanCentral SAST ignores it and informs you with a message.
`-sargs`, `--scan-args <scan_option>`	Specifies an OpenText SAST scan option (repeatable). For multiple scan options, use multiple `-sargs` options. If the scan option has a path parameter that includes a space, enclose the path in single quotes. For a list of OpenText SAST options you can use with the `-sargs` option, see Options accepted for -sargs (--scan-args).
`-sastver`, `--sast-version <version>`	Specifies the <year>.<quarter> OpenText SAST version to assign the remote translation and scan job. For more information about the supported OpenText SAST versions, see the *Application Security Software System Requirements* document.

1.10.3. Package command

Use the `package` command to create a ZIP archive (project package) of your project. You can either:

- Upload this project package to the Controller with the OpenText ScanCentral SAST`start` command
- Run an analysis with a locally installed OpenText SAST using the PackageScanner tool
- Upload this project package to OpenText Core Application Security for analysis

Caution

To avoid a packaging failure for projects with file paths that contain an umlaut, you must first add the `com.fortify.sca.CmdlineOptionsFileEncoding` property to the `<sast_install_dir>/Core/config/fortify-sca.properties` file and specify a value for it that is not ASCII encoding.

Package command option	Description
`-bt`, `--build-tool <name>`	Specifies the name of the build tool used for the project. The valid values for *<name>* are `dotnet`, `gradle`, `msbuild` (Windows only), `mvn`, and `none`. If not specified, OpenText ScanCentral SAST automatically detects the build tool based on the project files being scanned.
`-bc`, `--build-command <commands>`	(For use with Maven, Gradle, dotnet, and MSBuild) Specifies custom build parameters for preparing and building the project. The following example build command starts a Gradle build before packaging: `-Prelease=true clean customTask build` If you use the `-bc` option, and the build fails, OpenText ScanCentral SAST stops working on the build. (Gradle only) If you *do not* use `-bc`, the default command, default tasks, and target are invoked. If the build fails, OpenText ScanCentral SAST displays a warning, but continues to work and then displays a message to indicate that the build procedure failed and you might get incomplete results.
`-bf`, `--build-file <file>`	Specifies the build file if you are not using a default name such as `build.gradle` or `pom.xml`.
`-q`, `--quiet`	Prevents the printing of stdout from the build execution.
`-skipBuild`	Disables the project preparation build step if your projects use Gradle or Maven before packaging. If you use this option, any `-bc` option specified is ignored. If your project does not use a build tool, you can use this option to prevent OpenText ScanCentral SAST from automatically restoring dependencies using a package manager (for languages such as Go, JavaScript/TypeScript, PHP, and Python).
`-t`, `--include-test`	Includes the test source set (Gradle), the test scope (Maven), or projects in your solution that reference NUnit, xunit, or MSTest (.NET).
`-exclude <file_paths>`	Specifies files or directories (with absolute or relative path, or Ant-style path pattern) to exclude from a project package (repeatable). Separate multiple file paths with semicolons (Windows) or colons (Linux). For example, you might use this option to exclude a few test files from the project package.

`-include <file_paths>`	Specifies files or directories (with absolute or relative path, or Ant-style path pattern) to include in a project package (repeatable). Only file paths for files within the current working directory are included. Separate multiple file paths with semicolons (Windows) or colons (Linux). For example, you might use this option if you have only a few files you want to include in the project package. You can combine this option with the `-exclude` option to exclude specific files from the included path. For example commands, see Generating a OpenText ScanCentral SAST package.
`-hv`, `--php-version <version>`	Specifies the PHP version. If not specified, OpenText ScanCentral SAST automatically detects the installed PHP version.
`-oss`, `--open-source-scan`	(For use with OpenText Core Application Security only) Specifies to generate and collect additional files for open source software composition analysis. For details, see the *OpenText™ Core Application Security User Guide*.
`-sdu`, `--skip-debricked-update`	(For use with OpenText Core Application Security only) Specifies not to check for an updated version of the Debricked CLI. If this option is specified and no Debricked CLI is currently installed, then OpenText ScanCentral SAST generates and collects the additional files for open source software composition analysis without the Debricked CLI. You must also specify the `-oss` option to use this feature.
`-pyr`, `--python-requirements <file>`	Specifies the Python project requirements file to install and collect dependencies.
`-pyv`, `--python-virtual-env<dir>`	Specifies the Python virtual environment location.
`-yv`, `--python-version <version>`	Specifies the Python version to automatically find the installed Python. The valid values are `2` and `3`. This option is ignored if OpenText ScanCentral SAST client is started under a Python virtual environment or if `-python-virtual-env` is specified.
`-targs`, `--translation-args<option>`	Specifies an OpenText SAST translation option (repeatable) For multiple translation options, use multiple `-targs` options. If the translation option has a path parameter that includes a space, enclose the path in single quotes. For a list of OpenText SAST options you can use with the `-targs` option, see Options accepted for -targs (--translation-args).
`-o`, `--output <file>`	Specifies the output file name. The file extension must be `*.zip`. If not specified, OpenText ScanCentral SAST writes the project package to a ZIP archive with the name `fortifypackage.zip`.
`-dnr`, `--debricked-no-resolve`	Disables the Debricked resolve command that is automatically executed when the OpenText ScanCentral SAST client package command is run with the `-oss` option. Use this option if you want to prepare the Debricked files manually using the Debricked CLI directly, and to ensure that the OpenText ScanCentral SAST client does not overwrite these prepared files.

See also

Generating a ScanCentralSAST Package

Using the PackageScanner tool

1.10.4. Options accepted for -targs (--translation-args)

This topic lists the OpenText SAST translation options you can use with the OpenText ScanCentral SAST `-targs` option. You can use these options with the OpenText ScanCentral SAST `start` and `package` commands. For descriptions of the OpenText SAST translation options listed in this topic, see the *OpenText™ Static Application Security Testing User Guide*.

`-autoheap`
`-abap-includes`
`-appserver`
`-appserver-home`
`-appserver-version`
`-build-label`
`-build-project`
`-build-version`
`-checker-directives`
`-copydirs`
`-cp, -classpath`
`-debug`
`-debug-mem`
`-debug-verbose`
`-dialect`
`-disable-template-autodiscover`
	`-disable-language`
	`-django-disable-autodiscover`
	`-django-template-dirs`
	`-enable-language`
	`-encoding`
	`-exclude`
	`-extdirs`
	`-gotags`
	`-gopath`
	`-goproxy`
	`-jdk, -source`
	`-jinja-template-dirs`
	`-jvm-default`
	`-noextension-type`
	`-php-source-root`
		`-php-version`
		`-python-no-auto-root-calculation`
		`-python-path`
		`-python-version`
		`-quiet`
		`-ruby-path`
		`-rubygem-path`
		`-show-unresolved-symbols`
		`-source-base-dir`
		`-sourcepath`
		`-sql-language`
		`-v, -version`
		`-verbose`

1.10.5. Options accepted for -sargs (--scan-args)

This topic lists the OpenText SAST scan options you can use with the OpenText ScanCentral SAST `-sargs` option. You can use these options with the OpenText ScanCentral SAST `start` command. For descriptions of the OpenText SAST scan options listed in this topic, see the *OpenText™ Static Application Security Testing User Guide*.

`-analyzers`	`-disable-source-bundling`	`-no-default-rules`
`-autoheap`	`-disable-metatable`	`-no-default-sink-rules`
`-bin, -binary-name`	`-enable-analyzer`	`-no-default-source-rules`
`-build-label`	`-filter`	`-p, -scan-precision`
`-build-project`	`-fvdl-no-descriptions`	`-project-template`
`-build-version`	`-fvdl-no-enginedata`	`-quick`
`-debug`	`-fvdl-no-progdata`	`-quiet`
`-debug-mem`	`-fvdl-no-snippets`	`-rules`
`-debug-verbose`	`-legacy-jsp-dataflow`	`-sc, -scan-policy`
`-disable-analyzer`	`-machine-output`	`-v, -version`
`-disable-default-rule-type`	`-no-default-issue-rules`	`-verbose`

1.10.6. Status command

Use the `status` command to check the status of a remote scan job or the Controller.

Status command option	Description
`-ctrl`	Checks whether the Controller is running.
`-token`, `--job-token <token>`	Specifies the job token for a remote scan job.
`-bl`, `--block-until <action>`	Specifies to have the process (scan or merge) wait until the FPR upload and processing are complete, and then download the merged FPR file from Application Security. The following values are valid for *<action>*: • `scan`—Direct the scan process to continue to run until the scan is complete and available on the Controller. • `sscproc`—Wait for Application Security processing to complete. If the scan results file (FPR) is not uploaded to Application Security, an error occurs.
`-bto`, `--block-timeout <n>`	Specifies how long (in minutes) to block processing. The valid range for *<n>* is from `0` to `10080` minutes. Specify `0` for no timeout.
`-pi`, `--poll-interval <n>`	Specifies how frequently (in seconds) to poll the processing status. The valid range for *<n>* is from `10` to `60`.

See also

Viewing the Scan Request Status

1.10.7. Progress command

Use the `progress` command to get the progress of an OpenText SAST scan.

Important

If your projects are based on Java 11 or later, some sensor configuration is required to use the `progress` command . For instructions, see Configuring Sensors to Use the Progress Command when Starting on Java.

1.10.8. Retrieve command

Use the `retrieve` command to download the scan results, log files, and job file for a remote scan job from the OpenText ScanCentral SAST Controller.

Retrieve command option	Description
`-token`, `--job-token` *<token>*	Specifies the job token for a remote scan job.
`-f`, `--output-file` *<file>*`.fpr`	Specifies a file name for the local scan results (FPR) file.
`-j`, `--job-file` *<file>*`.zip`	Specifies a file name for the local job file that was submitted to OpenText ScanCentral SAST for analysis. The job file for remote translation contains the project package (sources, dependencies, and metadata). The job file for local translation contains the mobile build session (MBS) file.
`-log`, `--log-file` *<file>*	Specifies a file name for the local OpenText SAST log file.
`-slog`, `--sensor-log-file` *<file>*	Specifies the file name for the local sensor log output.
`-o`, `--overwrite`	Overwrites an existing scan results (FPR), log, or job file with new data.
`-block`	Specifies to wait for the job to complete and then download the scan results.
`-bto`, `--block-timeout` *<n>*	Specifies how long (in minutes) to block processing. The valid range for *<n>* is from `0` to `10080` minutes. Specify `0` for no timeout. The default value is `0`.
`-pi`, `--poll-interval` *<n>*	Specifies how frequently (in seconds) to poll the processing status. The valid range for *<n>* is `10` to `60` seconds.

See also

Retrieving Scan Results from the Controller

1.10.9. Upload command

Use the `upload` command to resend an FPR file to Application Security after a previous upload attempt failed.

Upload command option	Description
`-token`, `--job-token <token>`	Specifies the job token for the remote scan job to resend an FPR file to Application Security.

See also

Retrying Failed Uploads to Application Security

1.10.10. Cancel command

Use the `cancel` command to cancel a pending or running remote scan job.

Cancel option	Description
`-token`, `--job-token <token>`	Specifies the job token for the remote scan job you want to cancel.

See also

Canceling Scan Requests

1.10.11. Update command

Use the `update` command to update a client or sensor to the latest version available on the Controller. This updates a standalone client to the latest available client version. It updates an embedded client or sensor to the latest available patch version, but does not update them to the next major version.

Examples:

```
scancentral -url <controller_url> update
```

or

```
scancentral -sscurl <ssc_url> -ssctoken <token> update
```

1.10.12. Worker command

Use the `worker` command to assign a sensor pool or set a timeout.

Worker command option	**Description**
`-pool`, `--assign-to-pool` *<uuid>* | *<pool_name>*	Specifies the sensor pool to which the sensor is assigned after it connects to the Controller. If the sensor is already assigned to a pool, this option overrides that assignment. If an error occurs in the sensor pool assignment, the sensor shuts down. You can specify the sensor pool by either the UUID or the pool name.
`-sto`, `--scan-timeout` *<n>*	Specifies the maximum amount of time (in minutes) a sensor can work on an assigned job (and prevent the sensor from doing other work). Use of this worker option has a higher priority than the `scan_timeout` property setting in the `config.properties` file.