OpenText™ Core SAST Aviator

User Guide

Version : 26.1

PDF Generated on : January 16, 2026

Table of Contents

1. User Guide

This user guide provides guidance for users to use the OpenText™ Core SAST Aviator.

This document includes the following information:

- Key concepts of SAST Aviator (Introduction)
- Downloading and using the SAST Aviator (Getting started)
- List of administrator customer tasks that you can perform using SAST Aviator (Manage tenant information)
- List of queries for the customer administrator and user (FAQs)

1.1. Change log

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

Software Release / Document Version	Changes
26.1.0	Added: • New options to refresh the application version metrics and to set the refresh timeout while auditing (see Trigger audit) Updated: • All categories for JavaScript, TypeScript, Python, Go, and Kotlin are now supported with automatic suppression (see Supported languages and vulnerability categories)
25.4.0/Revision1: Dec 11,2025	Added: • Supports bulk auditing for fcli v3.13.1 and later (see Perform bulk audit)

Software Release / Document Version	Changes
25.4.0	Added: • Limitations for off-cloud and Fortify Hosted customers (see Limitations) • New command to download an FPR from OpenText Application Security and automatically apply auto-remediations suggested by SAST Aviator (see Perform auto-remediation) • New command options to select a filter set and folder or folders for auditing issues (see Trigger audit) • New command option to audit all issues by ignoring the filter sets including the default enabled one (see Trigger audit) • New command to add SAST Aviator tags to the OpenText Application Security filter templates, thus ensuring all the relevant templates are aligned with SAST Aviator tagging requirements (see Apply SAST Aviator tag)

Software Release / Document Version	Changes
25.3.0	Updated: • All categories for .NET are now supported with automatic suppression, except for Code Correctness, Dead Code, Poor Error Handling, and Unsafe Native Invoke (see Supported languages and vulnerability categories) • The email ID associated with managing user tokens is the user's email ID, which is other than the administrator's email ID (see Generate tokens for individual users, and Manage user tokens)
25.2.0	Initial release of the document.

1.2. Introduction

OpenText™ Core SAST Aviator is a cloud-based enterprise service that audits the OpenText SAST scan results as a true positive or a false positive.

OpenText™ Core SAST Aviator leverages Large Language Model (LLM) technology. When an issue is classified as a true positive, SAST Aviator offers remediation recommendations, enabling users to resolve code issues quickly and accurately. SAST Aviator also supports auto-remediation.

SAST Aviator is accessible using OpenText SAST in an off-cloud setup and OpenText SAST through the Fortify Hosted model. In both cases, SAST Aviator itself is a SaaS service. Regardless of the access method, you can use the open-source Fortify CLI tool to transmit scan results from the OpenText Application Security to SAST Aviator for processing. The results from SAST Aviator are stored as audit information in the OpenText Application Security. SAST Aviator is also available through Core Application Security (Fortify on Demand).

1.2.1. Product name changes

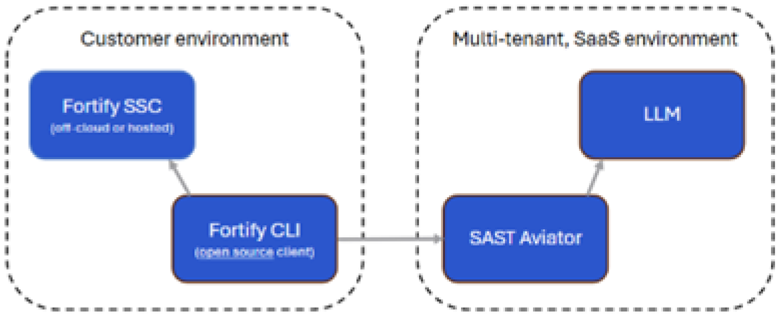
OpenText is in the process of changing the following product names:

Previous name	New name
Fortify Static Code Analyzer	OpenText™ Static Application Security Testing (OpenText SAST)
Fortify Software Security Center	OpenText™ Application Security
Fortify WebInspect	OpenText™ Dynamic Application Security Testing (OpenText DAST)
Fortify on Demand	OpenText™ Core Application Security
Debricked	OpenText™ Core Software Composition Analysis (OpenText Core SCA)
Fortify Applications and Tools	OpenText™ Application Security Tools

The product names have changed on product splash pages, mastheads, login pages, and other places where the product is identified. The name changes are intended to clarify product functionality and to better align the Fortify Software products with OpenText. In some cases, such as on the documentation title page, the old name might temporarily be included in parenthesis. You can expect to see more changes in future product releases.

1.2.2. SAST Aviator model

The Fortify CLI tool sends the SAST scan results, including vulnerability information and source code snippets, from the OpenText Application Security to SAST Aviator. The SAST Aviator sends the scan results to an LLM for auditing the scan results. The results are then returned to Fortify CLI. The vulnerability and audit information are not stored within SAST Aviator. SAST Aviator retains only the statistical data indicating that an audit occurred, including the number of issues identified.



Advantages of SAST Aviator

SAST Aviator leverages Generative Artificial Intelligence (GenAI) in the form of a Large Language Model (LLM) to generate content and provide product functionality.

- **Hybrid model**: The SAST Aviator model enables you to use the cloud service hosted by OpenText and the existing Fortify CLI utility instead of hosting a heavy LLM.
- **Integration with OpenText Application Security**: After a regular SAST scan is performed, the results are uploaded to the OpenText Application Security. Subsequently, SAST Aviator audits the scan results stored in OpenText Application Security. This approach prevents redundant evaluations.

1.2.3. Fortify CLI capabilities

In addition to auditing the scan results using fcli, you can perform the following operations:

- Apply the remediations proposed by SAST Aviator to the source code automatically.
- View entitlements and entitlement consumption. For more information, see Entitlement model.
- Create new applications and view available applications.
- Create access tokens for individual users.
- List and revoke the access tokens.

1.2.4. User roles

SAST Aviator includes sequential procedures and involves different user roles in your organization. The user roles involved are:

- Customer administrator: Customer administrator can create an admin configuration, create and manage tokens, and manage applications and entitlements.
- Customer user: User can create a user session and trigger an audit.

Note

Before using the fcli, you must be registered with OpenText. OpenText Support creates and registers the tenant upon your initial purchase of SAST Aviator.

1.2.5. Entitlement model

SAST Aviator is a paid service. The following entitlement models of SAST Aviator are available for purchase:The following entitlement models of SAST Aviator models are available for purchase:

- **per Application**: The basic entitlement model is **per Application**. In this model, you can run any number of audits for a single application. Therefore, the customer administrator must first create applications in the tenant. SAST Aviator monitors the number of entitlements for each tenant every time a Fortify Project Result (FPR) is processed.
- **per Developer**: In this model, SAST Aviator cannot monitor developers. By default, a maximum of four applications are allocated per 10 developers. If this allocation does not meet your requirements, contact your account representative.

For every new purchase of the SAST Aviator entitlement, OpenText Support updates the number of entitlements for the respective tenant.

1.2.5.1. Limitations

SAST Aviator, like any LLM-powered system, has usage limitations due to the operational costs of invoking LLMs. Each application is assigned with a quota that defines the maximum number of issues it can audit. Quotas apply individually to each application and are not shared across applications or tenants. This method provides an efficient way to control audit consumption. When an application reaches its quota, issues will not be audited until the quota is assigned.

Quota consumption

When you run an audit, SAST Aviator will audit up to the remaining quota for that application. If the audit is successful, the quota will be deducted based on the consumption. If the audit fails or crashes the quota remains unchanged.

Note

- Only one audit run per application is allowed at a time to ensure quota consistency.
- Quota usage is recorded in the audit logs for tracking.

Quota top-up

Initial Quota

When an application is created, an Initial Quota is credited as defined by the tenant configuration. The default value is 2500.

Monthly Quota

Every month, applications receive a Monthly Quota top-up. The default value is 500. The total quota for an application cannot exceed the Initial Quota.

For example:

- Initial Quota = 2500 (default)
- Monthly Quota = 500 (default)
- Month-1 = 300 issues are consumed and 2200 are remaining.
- Monthly top-up = 500 is the maximum, but the total quota is limited to the Initial Quota, which is 2500. Therefore, 300 is added (300 + 2200).

Note
In some cases, both Initial and Monthly Quotas can be configured based on the tenant. This change effects the future and not applied retroactively.

Behavior when quota limit is reached

If an audit reaches the quota limit:

- The number of issues audited is based on the remaining quota. FCLI displays the status and the number of issues audited and skipped.
- The audit stops gracefully.
- For the audited issues, the FPR file is generated and uploaded to OpenText Application Security.
- This case is not treated as an error.

When you run an audit and the quota is completely utilized, FCLI displays the message, "*No quota available for application <application_name>. Your next quota update for this application will happen on <Month Date, Year>.*".

Tip
The customer administrator can check the quota information using `fcli aviator app list`.

Efficient way to use the quota

When applications have a large number of issues exceeding the quota assigned and require auditing, use the following methods:

- Use OpenText SAST scan policy to reduce the number of issues. For information, see *OpenText™ Static Application Security Testing User Guide*.
- Use filter options while auditing:
    - Specify folders of your default filter set (for example, only audit Critical and High issues) to limit the number of issues processed by SAST Aviator, or,
    - Define a custom filter set specifically to manage what gets processed by SAST Aviator. For more information about the filter options, see Trigger audit.

See also

FAQs

1.2.6. Limitations (OpenText™ Core Application Security)

Important

The limitation system described below exclusively applies to SAST Aviator as available through OpenText™ Core Application Security (Fortify on Demand). For the limitation mechanism in SAST Aviator for off-cloud and Fortify Hosted customers, see Limitations.

There are limitations on how SAST Aviator audits SAST scan results containing extremely large number of issues.

SAST Aviator's approach to auditing data is similar to how auditing by a human works. If there are hundreds of issues, they can be audited manually. But if there are thousands, that’s neither practical nor useful. The first response should be to look for patterns. A recurring bad coding pattern may cause many true positives. In that case, the code should be fixed in bulk. Alternatively, some rules in SAST may trigger massive false positives for that particular codebase. In that case, the SAST scan configuration should be adjusted. After these steps, a smaller number of remaining findings can be audited individually.

SAST Aviator has largely been designed as an AI-powered version of a human auditor. It will not audit an unlimited number of issues. Practically, such a limit is also necessary, given the non-trivial resource consumption for every issue audited.

The following limits apply:

- For any FPR, SAST Aviator audits at most 2,500 new issues in total.
- For any FPR, SAST Aviator audits at most 500 new issues in a single category.

When SAST Aviator processes an FPR that exceeds one or more of these limits, any issue beyond the limit is marked as “Excluded due to Limiting”, and the specific limit is explained in the comment. Such issues are not audited in a subsequent run, either. When a subsequent SAST analysis of the same project yields new issues, these new issues are audited.

For example,

- A SAST scan of project X yields 3,000 issues.
- The SAST Aviator auditing will audit 2,500 of those, and mark 500 as “Excluded due to Limiting”.

- Now, development continues. An additional 100 issues have been found, leading to an FPR with 3,100 issues.
- SAST Aviator auditing of this new FPR will:
    - Not audit again the 2,500 previously audited issues.
    - Not audit the 500 issues previously marked as excluded.
    - Audit the 100 new issues.

1.2.7. Authentication model

SAST Aviator uses a dual authentication model.

- Customer administrators are authenticated using a private key. FCLI signs customer administrator requests with this private key. SAST Aviator verifies the signature with the registered public key. New customer administrators and public keys can be registered through OpenText support.
- Regular users are authenticated with an access token. Customer administrators can create these tokens for users. FCLI includes this access token in user requests.

1.2.8. Audit tag mapping

The SAST Aviator algorithm predicts issues to be true positives or false positives and, in some cases, unsure.

The OpenText Application Security needs to set an audit tag value, and decide to suppress an issue or not. As a user, you might have customized the available audit tag values in OpenText Application Security resulting in deviations from the default values. For these reasons, SAST Aviator has the functionality to map its predictions to audit tag values and suppression status in OpenText Application Security.

To configure this mapping, SAST Aviator considers the two tiers of support. For more information, see Supported languages and vulnerability categories. Considering there are two tiers and three different SAST Aviator outcomes (true positive (TP), false positive (FP), unsure), there are six different cases. These cases must be mapped to a OpenText Application Security audit value and a suppression status. The following is the default mapping performed by SAST Aviator:

Tier	Tier configuration name	Outcome	Audit value	Suppressed
Supported with automatic suppression	tier_1	TP	Exploitable	No
Supported with automatic suppression	tier_1	FP	Not an issue	Yes
Supported with automatic suppression	tier_1	Unsure	Not set	No
Supported without automatic suppression	tier_2	TP	Suspicious	No
Supported without automatic suppression	tier_2	FP	Not an issue	No
Supported without automatic suppression	tier_2	Unsure	Not set	No

To override the default tag mapping, use the `--tag-mapping` argument when you run an audit.

```
fcli aviator ssc audit --av <application_version_name:id> --tag-mapping=<file.yaml>
```

The following tag mapping file is the default one that implements the mapping as explained in the aforementioned table. Use this mapping file as a basis to configure your required mapping. In addition to changing audit tag values and suppression status, you can also select a different audit tag altogether.

```
 # Set the SSC tag to use to store Aviator results. Optional.
# If not set, defaults to "87f2364f-dcd4-49e6-861d-f8d3f351686b"
tag_id: "87f2364f-dcd4-49e6-861d-f8d3f351686b"


# Map Aviator results to SSC tag values. “tier_1” are issues that are
# high-confidence cases that by default are suppressed automatically.
# “tier_2” are the remaining issues.
# “value” is a String attribute that maps to a tag value in SSC. It may be
# omitted. In that case, Aviator will not set a value (but will still add a
# comment)
# “suppress” is a Boolean attribute that defaults to “false”


mapping:
  tier_1:
    fp:
      value: "Not an Issue"
      suppress: true
    tp:
      value: "Exploitable"
      suppress: false
    unsure:
      suppress: false
  tier_2:
    fp:
      value: "Not an Issue"
      suppress: false
    tp:
      value: "Suspicious"
      suppress: false
    unsure:
      suppress: false
```

1.2.9. 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.

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. **Note** This document is included only with the product download.
OpenText Application Security Software Release Notes	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.

Fortify ScanCentral SAST

The following document provides information about Fortify ScanCentral SAST. This document is available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-software-security-center.

Document / file name	Description
OpenText™ Fortify ScanCentral SAST Installation, Configuration, and Usage Guide sc-sast-ugd-*<version>*.pdf	This document provides information about how to install, configure, and use Fortify ScanCentral SAST to streamline the static code analysis process. It is written for anyone who intends to install, configure, or use Fortify ScanCentral SAST to offload the resource-intensive translation and scanning phases of their OpenText SAST process.

OpenText Application Security

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

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

OpenText Application Security Tools

The following documents provide information about OpenText Application Security Tools. These documents are available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools.

Document / file name	Description
OpenText™ Application Security Tools Guide sast-tgd-*<version>*.pdf	This document describes how to install application security tools. It provides an overview of the applications and command-line tools that enable you to scan your code with OpenText SAST, review analysis results, work with analysis results files, and more.
OpenText™ Fortify Audit Workbench User Guide awb-ugd-*<version>*.pdf	This document describes how to use Fortify Audit Workbench to scan software projects and audit analysis results. This guide also includes how to integrate with bug trackers, produce reports, and perform collaborative auditing.
OpenText™ Fortify Plugin for Eclipse User Guide ep-udg-*<version>*.pdf	This document provides information about how to install and use the Fortify Plugin for Eclipse to analyze and audit your code.
OpenText™ Fortify Analysis Plugin for IntelliJ IDEA and Android Studio User Guide iap-udg-*<version>*.pdf	This document describes how to install and use the Fortify Analysis Plugin for IntelliJ IDEA and Android Studio to analyze your code and optionally upload the results to OpenText Application Security.
OpenText™ Fortify Extension for Visual Studio User Guide vse-ugd-*<version>*.pdf	This document provides information about how to install and use the Fortify Extension for Visual Studio to analyze, audit, and remediate your code to resolve security-related issues in solutions and projects.

1.3. Supported languages and vulnerability categories

SAST Aviator is verified by OpenText to maximize accuracy. The extent to which a particular vulnerability category in a certain programming language is supported by SAST Aviator might differ based on the amount of verification and optimization that has already been performed. There are three classes:

Class	Description
Supported with automatic suppression	Cases with a high degree of confidence. By default, SAST Aviator performs automatic suppression of false positives.
Supported without automatic suppression	Cases where confidence is yet to be established to the same standard. By default, SAST Aviator does not perform automatic suppression.
Not supported	A small set of cases that cannot be handled by SAST Aviator.

The underlying LLM used by SAST Aviator evolves over time. Because not every LLM version is immediately available in all cloud hosting locations used by SAST Aviator, different instances of SAST Aviator may use different LLM versions at any point. The LLM version in use determines the classification of cases. Generally, on newer LLMs, more classes can be moved to “automatic suppression”.

The following overview lists how language or category combinations are classified in the current version of SAST Aviator for off-cloud and hosted customers.

Supported language or category combinations with automatic suppression

- Java, JavaScript, TypeScript, Python, Go, Kotlin
    - All categories except explicitly non-supported ones.
- .NET

    - All categories except Code Correctness, Dead Code, Poor Error Handling, and Unsafe Native Invoke.

    Note

    The categories Code Correctness, Dead Code, Poor Error Handling, and Unsafe Native Invoke are "supported without automatic suppression".

Supported without automatic suppression

- All other language or category combinations supported by OpenText SAST, except explicitly excluded cases.

Not supported

- The following vulnerability category is explicitly not-supported:
    - Privilege Management: Unnecessary Permission.

Note

The verification of this category's issues requires access to the complete source code at once, which is not compatible with the way SAST Aviator functions.

1.4. Getting started

Perform the following steps to get started with SAST Aviator:

1. Download fcli
2. Register customer administrator
3. Generate tokens for individual users
4. Create application
5. Apply SAST Aviator tag
6. Trigger audit / Perform bulk audit
7. Perform auto-remediation

1.4.1. Download fcli

Download fcli v3.14.2 or later from https://github.com/fortify/fcli and extract it into a directory from where you will run the commands.

1.4.2. Register the customer administrator

Prerequisite

Ensure that the tenant is registered. Contact OpenText Support for details.

Register the customer administrator

To register a customer administrator:

1. Create a 4096-bit RSA key pair in the PEM format using a cryptographic tool available in your organization.
    - (Optional) Use OpenSSL (https://www.openssl.org/) to generate the RSA key pair.
        1. Generate a 4096-bit RSA private key (private_key.pem):

           ```
           openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096
           -out private_key.pem
           ```

        2. Extract the public key (public_key.pem) from the private key:

           ```
           openssl rsa -in private_key.pem -pubout -out public_key.pem
           ```

2. Save the private key (private_key.pem) for later use.
3. Share the public key (public_key.pem) and other necessary details with OpenText Support to register the customer administrator.

   You will be notified after the registration is complete.

1.4.3. Generate tokens for individual users

Prerequisite

You must be a registered customer administrator.

Generate tokens for individual users

To create an access token for a specified user:

1. At the command prompt, navigate to the path where fcli has been extracted.
2. (Optional) View the various administrator and user operations available:

   ```
   fcli aviator -h
   ```

   The following table lists the available command options:

Argument	Description
`-h, --help`	Shows the help message and exits.
`admin-config`	Manages the SAST Aviator administrator configurations (start here).
`session`	Manages the SAST Aviator user sessions (start here).
`app`	Manages the SAST Aviator applications.
`entitlement`	Manages the SAST Aviator entitlements.
`ssc`	Uses SAST Aviator with OpenText™ Application Security.
`token`	Manages SAST Aviator access tokens.

> **Note**
>
> Use **admin-config** to view entitlement, manage applications, and generate tokens and **session** to audit.
> Ensure to create an **admin-config** before performing administrator tasks and a user **session** before auditing.

3. Create an administrator configuration for interacting with SAST Aviator:

```
fcli aviator admin-config create --url <aviator_server_url> --tenant <tenant_name> --private-key <path_to_private_key.pem>
```

> **Note**
>
> The `--private-key` can be a file containing the key or the key itself. Ensure that the key is in the PEM format.

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

An admin session is created.

4. Generate the user access token:

```
fcli aviator token create --email <user_email_id> --save-token <output_file>
```

Optional arguments	Description	Default value
`--name`	Specifies the token name.	Derived from `--email`
`--save-token`	Saves the generated raw token string to the specified file. By default, the string is in json format.	NA
`-o, --output`	Specifies the token format. The available formats are csv, table, expr, json, xml, and yaml.	NA
`--to-file`	Specifies a file to save the output.	NA
`--end-date`	Specifies the token expiration date in the YYYY-MM-DD format.	30 days from the date of creation

Note

OpenText recommends using the `--save-token` optional argument to ease the user experience when using the access token.

1.4.4. Create application

Prerequisites

- You must be a registered customer administrator.
- Ensure to have a valid entitlement. See Entitlement model section.

Create application

To create an application:

1. Create an administrator configuration for interacting with SAST Aviator

   ```
   fcli aviator admin-config create --url <aviator_server_url> --tenant
   <tenant_name> --private-key <path_to_private_key.pem>
   ```

   Note

   The `--private-key` can be a file containing the key or the key itself. Ensure that the key is in the PEM format.

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

2. Create an application:

   ```
   fcli aviator app create <aviator_application_name>
   ```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

   An application is created and assigned to the first available entitlement.

1.4.5. Apply SAST Aviator tag

If the OpenText Application Security application does not have the SAST Aviator specific tags applied, apply the SAST Aviator tags to the OpenText Application Security filter templates.

Apply tag

To apply SAST Aviator specific tags:

1. Log in to your Fortify Software Security Center session:

```
fcli ssc session login --url <ssc_url> -u <user_name> -p <ssc_password>
```

2. Apply the required template. Ensure to specify at least one option:

Note
Applying the SAST Aviator tags ensures that the SAST Aviator specific custom tags exists in OpenText Application Security and is associated with the specified issue templates and/or application versions. This prevents OpenText Application Security from removing the SAST Aviator audit data when processing the uploaded FPR files.

```
fcli aviator ssc prepare --all-avs | --all-issue-templates | --av=
<specific_app_VersionNameOrId> | --appversion=
<specific_app_VersionNameOrId> | --issue-template=
<specific_app_templateNameOrId>
```

Optional argument	Description	Default value
`--ssc-session`	Name of the SSC session.	default

Note
This command may change in a future fcli version to re-use generic tag update functionality that is planned for the `fcli ssc` module.

1.4.6. Trigger audit

The instructions provided in this topic are for auditing a single application. If you want to audit all applications in OpenText Application Security, or a subset of applications having a certain attribute, see the Perform bulk audit section.

Prerequisites

- You must be a customer user.
- Ensure the following:
  - At least one application is assigned to the entitlement.
  - You have a valid user access token.
- Logged in to your OpenText Application Security session.

Trigger audit

Caution

For the SAST Aviator to audit FPRs and provide remediation suggestions, ensure the following:

- Source code is available in the FPRs.
- Do not enable the option `-disable-source-bundling` or property `com.fortify.sca.FPRDisableSourceBundling` when you perform the scan through OpenText SAST.

To trigger an audit:

1. Create a user session to interact with SAST Aviator:

```
fcli aviator session login --url <aviator_server_url> --token <access_token>
```

Note

The default value for `--token` is a file path. To use other formats for the access token, prefix the value with `file:<local file containing key>` or `string:<key string value>` or `env:<env-var name containing key>`.

Ensure to create a user session before auditing.

If you cannot locate your access token, contact your customer administrator.

Optional argument	Description	Default value
--av-session, --aviator-session	Name of the Aviator user session.	default

2. Audit the application:
   This command downloads the FPR from the specified OpenText Application Security application version, SAST Aviator audits the issues and uploads the result back to the OpenText Application Security.

   ```
   fcli aviator ssc audit --av <application_version_name:id>
   ```

Optional arguments	Description	Default value
`--app`	Name of the Aviator application. If the name is not specified, the build ID of the FPR is considered.	FPR build ID
`--tag-mapping`	Overrides the default tag mapping using the YAML file. See Audit tag mapping.	tag mapping.yaml
`--ssc-session`	Name of the SSC session to use for auditing.	default
`--no-filterset`	Do not apply any filter sets, including the default enabled filter set from the FPR.	-
`--av-session, --aviator-session`	Name of the Aviator user session.	default
`--[no-]refresh`	By default, this option refreshes the source application version's metrics when copying from it. Note that for large applications, this can lead to an error if the timeout period expires.	-

Optional arguments	Description	Default value
`--refresh-timeout`	Set the timeout period for refreshing the metric. For example, 30s (30 seconds), 5m (5 minutes), 1h (1 hour).	60 seconds

It might take a few minutes to process the FPR. The duration depends on the size of the FPR.

Important

If the system running the SAST Aviator scan does not have sufficient physical memory, the audit may fail with an *OutOfMemoryError*. This error typically occurs when the available system memory or the configured JVM heap size is insufficient to process a huge number of vulnerabilities (FPR files containing thousands of issues) during the audit phase.

In such cases, ensure that the machine initiating the SAST Aviator scan has adequate physical memory and increase the JVM heap allocation before re-running the audit.

Note

- You can use the same access token to audit multiple FPRs on different terminals at the same time.
- You can audit an FPR only once.

1. You can use the following optional methods to audit:
   - Select a filter set for auditing:

     ```
     fcli aviator ssc audit <app_version_name_or_id> --filterset <filterset_name_or_id>
     ```

Optional argument	Description	Default value
`--filterset`	Specifies the name or ID of the filter set to apply.	default

- Select folder or folders for auditing:

```
fcli aviator ssc audit <app_version_name_or_id> --filterset
<filterset_name_or_id> --folder <folder_name1>,
<folder_name2>,...
```

Optional argument	Description	Default value
`--folder`	Filters the issues by a comma-separated list of specific folder names from the selected filter set (for example, Critical, High). This option requires an active filter set.	-

After the SAST Aviator processes the FPR, the Action status and the number of audited issues are displayed. SAST Aviator uploads the audited FPR back to the OpenText Application Security application version.

To view the SAST Aviator's audit results:

1. Open the OpenText Application Security application and go to **Applications** > **Audit**.
2. Click each row to view the Audit details, such as analysis tag, remediation comment, and the highlighted vulnerable code segment.

1.4.7. Perform bulk audit

Important

If the system running the SAST Aviator scan does not have sufficient physical memory, the audit may fail with an *OutOfMemoryError*. This error typically occurs when the available system memory or the configured JVM heap size is insufficient to process a huge number of vulnerabilities (FPR files containing thousands of issues) during the audit phase.

In such cases, ensure that the machine initiating the SAST Aviator scan has adequate physical memory and increase the JVM heap allocation before re-running the audit.

The bulk audit option is available for the fcli v3.13.1 and later. It helps to audit multiple projects in OpenText Application Security that have unaudited issues using a single command.

The bulk audit task identifies OpenText Application Security application versions with pending audit issues and automatically submits them to SAST Aviator for auditing. The action filters to only process versions with actual pending audit issues, automatically ignoring versions for which audit information needs to be refreshed.

For application versions that are not available in SAST Aviator, the action will automatically create them before submitting for audit.

You must specify either `--tag-mapping` or `--add-aviator-tags` option. The default tag mapping does not audit unsure issues, causing indefinite reselection. When `--tag-mapping` is specified, tags are required for all the values in the .yaml file, for example, for Tier_1 and Tier_2, all values FP, TP, and Unsure should have the tags correctly specified, else FPRs will be reaudited.

When `--add-aviator-tags` is specified, fcli runs `fcli aviator prepare` command for each application before auditing. This option automatically ensures that the SAST Aviator specific custom tags exist in OpenText Application Security and is associated with the specified issue templates and/or application versions.

Prerequisites

Ensure the following:

- SAST Aviator administrator configuration is created
- Active OpenText Application Security session
- User session is created

Trigger bulk audit

To perform bulk auditing for OpenText Application Security, run the following command:

```
fcli ssc action run bulkaudit --add-aviator-tags
```

Optional argument	Description	Default value
`--max-audits` , `-m`	Maximum number of project versions to audit.	-1 (unlimited)
`--filter` , `-f`	An optional filter to apply when querying Fortify Software Security Center for projects. For example, 'Languages:java'. For information on how to use this option, see the Usage of --filter section.	no filtering
`--exclude-filter` , `-e`	An optional inverse filter to exclude projects from audit. Projects matching this filter will be skipped. For example, 'Languages:c#' to exclude .NET projects. This can be combined with `--filter` .	no exclusion
`--dry-run` , `-n`	Shows what SAST Aviator commands will be run without actually running them.	false

Optional argument	Description	Default value
`--tag-mapping`, `-t`	The path to tag mapping YAML file. This custom mapping ensures that SAST Aviator's 'Unsure' audit results are properly handled. **Note** • Use this option if you do not use `--add-aviator-tags`. • Ensure to use either `--tag-mapping` or `--add-aviator-tags` when you run the `bulkaudit`.	

Optional argument	Description	Default value
`--add-aviator-tags`	If specified, fcli runs `fcli aviator prepare` for the application before auditing. **Note** • Use this option if you do not use `--tag-mapping`. • Ensure to use either `--tag-mapping` or `--add-aviator-tags` when you run the `bulkaudit`.	
`--refresh`	Refreshes application version metrics before audit. For large applications, this can lead to timeout errors.	true

Optional argument	Description	Default value
`--refresh-timeout`	Timeout period for metric refresh. For example, 30s (30 seconds), 5m (5 minutes), 1h (1 hour).	60s
`--log-file`	Saves the log output to a file. It is a good practice to use this option. You can refer to the log file for the complete information corresponding to the triggered `bulkaudit`.	./fcli.log

Note
Use either the `--tag-mapping` or `--add-aviator-tags` option, else an error is returned.

Usage of `--filter`

The `--filter` option in fcli enables you to perform a bulk audit by targeting specific attributes in your OpenText Application Security application.

For default attributes

1. Refer to your OpenText Application Security application to obtain the corresponding value for the default attribute that you want to audit.
2. Run bulk audit:

   ```
   fcli ssc action run bulkaudit --filter <default_tag_attribute:value> --add-aviator-tags
   ```

   For example:

   ```
   fcli ssc action run bulkaudit --filter Languages:ABAP --add-aviator-tags
   ```

For custom attributes

1. List the OpenText Application Security attributes:
   This displays details, such as ID, Category, Guid, Name, Type, and Required.

   ```
   fcli ssc attribute lsd
   ```

2. Get the Guid information of the required custom tag attribute:
   This command displays the related options, which contain ID, guid, name, description, hidden, inUse, index, projectMetaDataDefId, publishVersion, and objectVersion.

   ```
   fcli ssc attribute get-definition <custom_tag_attribute_guid_value>
   ```

3. Run the bulk audit:

   ```
   fcli ssc action run bulkaudit --filter
   <custom_tag_attribute_guid_value:required_guid_value> --add-aviator-tags
   ```

1.4.8. Perform auto-remediation

SAST Aviator allows users to download the audited FPR file from the OpenText Application Security artifact ID and automatically apply the remediations proposed by SAST Aviator to the source code with a single command. This enables users to fix their source code without any manual intervention.

Prerequisites

Ensure the following:

- Have an active user session.
- Already performed the SAST Aviator audit.

Apply the auto-remediation

To apply the SAST Aviator's remediation for OpenText Application Security:

1. Get the application version artifact information:

```
fcli ssc artifact list --av <application_versionnameorID>
```

Optional argument	Description	Default value
`--ssc-session`	Name of the OpenText Application Security session.	default

2. Apply auto-remediation:
   This command downloads the FPR from the OpenText Application Security artifact ID and automatically applies the remediations proposed by SAST Aviator on the source code directory.

```
fcli aviator ssc apply-remediations --artifactId <aviatorArtifactID>
```

Optional argument	Description	Default value
`--ssc-session`	Name of the OpenText Application Security session.	default
`--source-dir`	Specifies the source code directory. The remediations are applied to this directory. If the directory is not specified, remediations are applied to the current working directory.	Current working directory

After the remediations are applied, the total number of remediations, number of remediations applied and skipped, and the action status are displayed.

Note
If the source code file is still exactly the same from the time when it was scanned by OpenText SAST and then audited by SAST Aviator, the remediations are implemented. If the source code file has been changed in the meantime, remediations are implemented only if the relevant context is located accurately else the remediations are skipped.

1.5. Manage tenant information

Prerequisite

You must be a registered customer administrator.

List of customer administrator tasks

You can perform the following administrator tasks using the SAST Aviator fcli:

- Manage administrator configurations
    - Create an administrator configuration.
    - List the administrator configurations.
    - Delete an administrator configuration.
- Manage user tokens
    - Create a user access token.
    - List the access tokens.
    - Validate, revoke, or delete the access tokens.
- Manage applications
    - Create an application.
    - Get the application details.
    - List the available applications.
    - Update or delete the applications.
- Manage entitlements
    - Retrieve the entitlement details for a specified tenant.

Note

You must create an **admin-config** to interact with SAST Aviator before performing an administrator task.

1.5.1. Manage administrator configurations

You can perform the following tasks using the **fcli aviator admin-config** command.

- Create an administrator configuration for interacting with SAST Aviator:

```
fcli aviator admin-config create --url <aviator_server_url> --tenant
<tenant_name> --private-key <path_to_private_key.pem>
```

Note

The `--private-key` can be a file containing the key or the key itself. Ensure that the key is in the PEM format.

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- List the available SAST Aviator administrator configurations:

```
fcli aviator admin-config list
```

- Delete the SAST Aviator administrator configuration:

```
fcli aviator admin-config delete --admin-config
<administrator_configuration_name>
```

1.5.2. Manage user tokens

You can perform the following tasks using the **fcli aviator token** command.

- Create an access token for a user:

```
fcli aviator token create --email <user_email_id> --save-token <output_file>
```

Optional arguments	Description	Default value
`--name`	Specifies the token name.	Derived from `--email`
`--save-token`	Saves the generated raw token string to the specified file. By default, the string is in json format.	NA
`-o, --output`	Specifies the token format. The available formats are csv, table, expr, json, xml, and yaml.	NA
`--to-file`	Specifies a file to save the output.	NA
`--end-date`	Specifies the token expiration date in YYYY-MM-DD format.	30 days from the date of creation

Note

OpenText recommends using `--save-token` optional argument to ease the user experience when using the access token.

- Retrieve the list of access tokens within an Aviator tenant:

```
fcli aviator token list
```

- Retrieve a list of access tokens for a specified user within an Aviator tenant:

  ```
  fcli aviator token list --email <user_email_id>
  ```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- Validate an existing access token for a user within an Aviator tenant, checking its authenticity and status:

```
fcli aviator token validate --token <access_token>
```

Note

The default value for `--token` is a file path. To use other formats for the access token, prefix the value with `file:<local file containing key>` or `string:<key string value>` or `env:<env-var name containing key>`.

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- Revoke an access token for a user within an Aviator tenant. This task invalidates the access token without permanently deleting it:

```
fcli aviator token revoke --token <access_token>
```

Note

The default value for `--token` is a file path. To use other formats for the access token, prefix the value with `file:<local file containing key>` or `string:<key string value>` or `env:<env-var name containing key>`.

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- Delete an existing access token for a user:

```
fcli aviator token delete --token <access_token>
```

Note

The default value for `--token` is a file path. To use other formats for the access token, prefix the value with `file:<local file containing key>` or `string:<key string value>` or `env:<env-var name containing key>`.

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

1.5.3. Manage applications

You can perform the following tasks using the **fcli aviator app** command.

- Create applications:

```
fcli aviator app create <aviator_application_name>
```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- Get the details of an existing Aviator application for a particular tenant:

```
fcli aviator app get <application_id>
```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- List Aviator applications for a particular tenant:

```
fcli aviator app list
```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- Rename an existing SAST Aviator application identified by its ID:

```
fcli aviator app update <application_id> -n <new_application_name>
```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

- Delete an Aviator application by its ID:

```
fcli aviator app delete <application_id>
```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

1.5.4. Manage entitlements

Retrieve the entitlement details for a specified tenant in SAST Aviator:

```
fcli aviator entitlement list
```

Optional argument	Description	Default value
`--admin-config`	Name of the Aviator administrator configuration.	default

The following entitlement details are retrieved:

- Total number of entitlements available.
- Number of active entitlements.
- Total number of applications linked to the entitlements that are already consumed.
- Number of remaining applications under the available entitlements.
- Expiration date for each entitlement.

1.6. FAQs

What should you do if you disagree with the SAST Aviator audit result?

OpenText recommends creating a support request and sharing the FPR.

What should you do if you cannot locate your access token?

Contact your customer administrator.

What should you do if you run out of entitlements?

Contact your account representative.

What happens if your audit exceeds the quota?

The audit will stop once the quota is reached. The FPR will still be generated and uploaded, but further issues will not be audited.

Will quota changes affect past audits after configuring new Initial and Monthly quotas?

No, changes apply only to future audits and quota top-ups.

How to check your remaining quota?

Quota usage is visible in the audit logs for each application or contact your customer administrator.