Oracle® Communications Solution Test Automation Platform

Security Guide

Release 1.25.1.0.0
G23297-02
May 2025

Oracle Communications Solution Test Automation Platform Security Guide, Release 1.25.1.0.0

G23297-02

Contents

Preface

This guide provides guidelines and recommendations for managing security in Oracle Communications Solution Testing Automation Platform.

Audience

This guide is intended for system administrators, database administrators, and developers.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1 Overview

Learn about the Oracle Communications Solution Test Automation Platform (STAP) security.

Topics in this chapter:

Basic Security Considerations

The following principles are essential for ensuring the secure use of any application:

- **Keep software up-to-date:** This includes the latest product release and any patches that apply to it.
- **Limit user access or privileges:** Users should be given only the access necessary to perform their work. User privileges should be reviewed periodically to determine relevance to current work requirements.
- **Monitor system activity:** Establish who should access which system components, and how often, and monitor those components.
- **Install software securely:** For example, use firewalls, secure protocols (such as Transport Layer Security (TLS) and Secure Sockets Layer (SSL), and secure passwords). See "Securing STAP Deployment" for more information.
- **Learn about secure coding guidelines:** Secure-Text Encryption Tool employs AES-192 for encryption and decryption, creating unique keys and initialization vectors for each operation. It ensures the protection of confidential information, such as passwords and OAuth details, in environment configuration files. See "Secure Coding Guidelines for STAP" for more information.
- **Ensure secure STAP deployment:** Follow the necessary steps to ensure secure deployment for STAP. See "Ensuring a secure STAP deployment" for more information.
- **Keep up to date on security information:** Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible. See the Critical Patch Updates and Security Alerts website:

  `http://www.oracle.com/technetwork/topics/security/alerts-086861.html`

Overview of STAP Security

STAP security is designed to protect product, account, order, and asset data, as well as logs, and interfaces.

- **Application security:** Access to application modules and artifacts is authenticated using Basic Auth/Oauth.
- **Data security:** Scenarios, environment data, user information, and reports are secured in an encrypted database.

- **Interface security:** STAP composite service and references (interfaces) are secured by WebLogic Server security policies using Web Services Manager (WSM). Credentials for accessing external systems are configured and stored securely.

Understanding the STAP Environment

When planning your Oracle STAP implementation, consider the following:

- **Which resources need to be protected?**
  - Customer data, such as credit card numbers.
  - Internal data, such as confidential proprietary source code.
  - System components from being disabled by external attacks or intentional system overloads.
- **Who are you protecting data from?**

  Oracle recommends that you do not use any real-world data with STAP. Always use test data which can be generated from STAP or a supported external text generation tool. You can analyze your workflows to determine who needs access to the data; for example, it is possible that a system administrator can manage your system components without needing to access the system data.
- **What will happen if protections on strategic resources fail?**

  In some cases, a fault in your security scheme is merely an inconvenience. In other cases, it might cause significant damage to you or your customers. Understanding the security implications of each resource will help you protect it properly.

Restricting Permissions for Oracle STAP Directories

Oracle recommends keeping the permissions as restrictive as possible for your business needs. When installing on UNIX or Linux, consider using **umask 066** to deny read and write permission to all users except the user who installed the software. Table 1-1 lists the directories in which Oracle STAP creates files. Examine these directories to ensure they have the appropriate permissions.

Table 1-1 Oracle STAP Directories

Name	Description
Fusion Middleware home	The directory in which Oracle Fusion Middleware components are installed. This directory contains the base directory for Oracle WebLogic Server, among other files and directories.
Oracle STAP home (COMMS_HOME environment variable)	The directory in which Oracle STAP is installed. This is the `comms_home` directory within the Oracle base directory.
Domain home	The directory that contains the configuration for the domain onto which Oracle STAP is deployed. The default is *MW_home*/**user_projects/domains/***domain_name* (where *MW_home* is the Fusion Middleware home and *domain_name* is the name of the Oracle STAP domain), but it is frequently set to some other directory at installation.

Port Security

STAP communicates over a limited number of ports. Depending on your solution requirements, additional ports may be required, especially if Oracle STAP is deployed to a WebLogic Server cluster.

Table 1-2 lists the types of ports Oracle STAP uses.

Table 1-2 Oracle STAP Ports

Port	Port Description
Administration server port	The default value is 7001, but a different value can be set during domain creation.
Administration server SSL port	The default value is 7002, but a different value can be set during domain creation.
Node Manager port	The default value is 5556, but a different value can be set during Node Manager configuration.
SOA managed server ports	The default value is 8001, but a different value can be set during domain creation. In a clustered deployment, each managed server should have a different port. For example, 8002, 8003, and so on.
Oracle HTTP Server port	The default value is 7777, but a different value can be set during Oracle HTTP Server configuration.
SOA database port	The default is 1521, but a different value can be set during database creation.

2

Secure Coding Guidelines for STAP

Learn how the Secure-Text Encryption Tool employs AES-192 for encrypting, decrypting, and creating unique keys and Initialization Vectors (IVs) for each operation in Oracle Communications Solution Test Automation Platform (STAP).

Topics in this chapter:

Secure Plain Passwords in Configuration Files

Secure Text Encryption Tool is a secure data encryption utility that uses the Advanced Encryption Standard (AES) algorithm with a 192-bit key length to encrypt and decrypt data. It generates a random secret key and IV for each encryption operation, ensuring that the encrypted data is unique and secure. For the decryption operation, the same secret key is used and the IV is extracted from the encrypted text.

Technical Details

- **Encryption Algorithm:** AES (Rijndael)
- **Key Size:** 192-bit
- **Mode of Operation:** CBC (Cipher Block Chaining)
- **IV:** A random IV is generated and prepended to the encrypted data. The IV must be used during decryption alongside the secret key.

Generating a Secret Key Using keytool

```
keytool -genseckey -keystore "filename.jceks" -storetype jceks
storepassPhrase -keyalg AES -keysize 192 -alias aliasName
```

where:

- *filename* is file name for your keystore.
- *storepassPhrase*, if included, is the phrase to access the keystore password. The format for this can be either of the following:
  - **-storepass:env** *envVar* (with *envVar* being the name of an environment variable that contains the keystore password)
  - **-storepass:file** *pwFile* (with *pwFile* being the name of a file that contains the keystore password)

If you do not include this phrase, you will be prompted for the keystore password.

- *aliasName* is the name of the alias for the secret.

This command generates a 192-bit AES secret key and stores it in a JCE keystore file called **keys.jceks**.

You can read the JCEKS keys from your keystore using the following command:

```
keytool -list -keystore "filename.jceks" -storetype jceks storepassPhrase
```

Encryption and Decryption

- The tool generates a random 192-bit AES secret key and a random IV.
- The plaintext data is encrypted using the AES algorithm with the generated secret key and IV.
- The encrypted data is prepended with the IV, forming the final encrypted output.
- The IV is extracted from the beginning of the encrypted data.
- The remaining ciphertext is decrypted using the AES algorithm with the provided secret key and the extracted IV.
- The decrypted plaintext data is returned.

Using the security key password

The tool allows you to encrypt sensitive data within a properties file using the AES encryption algorithm. Here's a breakdown of the process:

1. Set the command line arguments:

- **.properties filepath:** Path to the properties file containing data to be encrypted.
- **.jceks filepath:** Path to the Java Key Store (JCEKS) file used to store the secret key.
- **Keystore Password:** Password to access the JCEKS keystore.
- **Alias Name:** The alias name within the JCEKS keystore that identifies the secret key.

2. Property Identification:

The tool scans the specified properties file for entries where the value equals "${SECURE_PASSWORD}". This indicates properties containing sensitive data.

3. User Interaction:

For each identified property, you are prompted to enter a new password.

4. Encrypt and Update:

- The new password is encrypted using the generated secret key.
- The value of the identified property is replaced with the following format:

```
${SECURE_TEXT("new-password")}
```

The original properties file is updated with the new placeholders indicating encrypted data.

> **Note:**
>
> This approach stores the encrypted passwords within the JCEKS keystore, requiring the keystore and password for decryption. Ensure the JCEKS keystore is protected with a strong password and stored securely.

Security Considerations

- **Key Management**: It is crucial to keep the secret key secure and protect it from unauthorized access. The security of the encrypted data relies entirely on the confidentiality of the secret key.
- **IV**: The IV is not a secret value and should be shared along with the encrypted data for successful decryption. Therefore, it is prepended to the encrypted data during the encryption process.

Secure Passwords in Configuration (Environment) Files

To secure sensitive data like passwords and OAuth credentials within your application's environment configuration files:

1. Generate a secret key by running the following:

   ```
   keytool -genseckey -keystore "filename.jceks" -storetype jceks
   storepassPhrase -keyalg AES -keysize 192 -alias aliasName
   ```

   where:

   - *filename* is file name for your keystore.
   - *storepassPhrase*, if included, is the phrase to access the keystore password. The format for this can be either of the following:
     - **-storepass:env** *envVar* (with *envVar* being the name of an environment variable that contains the keystore password)
     - **-storepass:file** *pwFile* (with *pwFile* being the name of the file that contains the keystore password)

     If you do not include this phrase, you will be prompted for the keystore password.
   - aliasName is the name of the alias for the secret.

   This command generates a 192-bit AES secret key and stores it in a JCE keystore file called **keys.jceks**.
2. You can read the JCEKS keys from your keystore using the following command:

   ```
   keytool -list -keystore "filename.jceks" -storetype jceks storepassPhrase
   ```

3. Run the encryption tool by running the following command:

   ```
   ./gradlew encryptPasswords -PtaasArgs="['file-path','keyfile-
   path','keystore-pass','alias-name']"
   ```

   Where:

   - *file-path* is the path to your `.properties` file containing the data to be encrypted.
   - *keyfile-path* the path to the JCEKS file (`filename.jceks`) storing the secret key.
   - *keystore-pass* is the keystore password.
   - *alias-name* is the alias name identifying the secret key.
4. Enter your new password.

For each `${SECURE_PASSWORD}` entry, you will be asked to enter a new password.

5. Encrypt and update your password.
   - The tool encrypts the new password using the secret key.
   - It replaces the original `${SECURE_PASSWORD}` entry with `${SECURE_TEXT("new-password")}`.

   Your original properties file is updated with encrypted passwords.

3
Securing STAP Deployment

Learn how the Oracle Communications Solution Test Automation Platform (STAP) ensures secure deployment and protection of confidential information.

Topic in this chapter:

Ensuring a Secure STAP Deployment

It emphasizes the importance of general security guidelines, such as access management to the environment, and OS and network-level security, to prevent unauthorized access to the deployment environment. Ensure to secure the deployment by running it only after safeguarding confidential information. To ensure a secure deployment for STAP, you need:

- **Kubernetes Secrets:** Kubernetes uses secrets to store confidential information. Use the scripts from the STAP Cloud Network Toolkit (CNTK) to ensure all passwords are protected.

  See "Installing STAP MIcro-Services" in *STAP Deployment Guide* for more information.
- **SSL certificate:** Use SSL certificate to secure communication between STAP components. See *STAP Deployment Guide* for details on creating and using SSL certificates during STAP deployment.
- **Environment Access:** Since STAP is a platform targeted for testing your solutions, ensure that STAP does not have access to any production environments. Do not use STAP to connect to any other systems other than the ones being used for testing.
- **OAuth:** STAP can be used with either Basic Auth or OAuth. For a more secure deployment, STAP uses OAuth to secure access to its UI.

STAP supports OAuth only when used in conjunction with IDCS. Ensure that you secure access to STAP for IDCS users.

4

Securing Your STAP Deployment

Learn about security considerations for your Oracle Communications Solution Test Automation Platform (STAP) deployment.

Based on the variety of customizations and plugins you have for your Kubernetes platform, you need to consider all possible security risks and have a mitigation plan in place.

Topic in this chapter:

General Security Considerations

Consider the following general security guidelines:

- Because the override **values.yaml** file for the Helm charts can be stored in versioning systems, it is recommended that you do not use it to save sensitive information, such as application credentials. Instead, use Kubernetes secrets.
- Use the sample scripts provided with the cloud native toolkit for creating secrets to maintain credentials for various applications, such as Order and Service Management (OSM), Siebel, Billing and Revenue Management (BRM), Service-Oriented Architecture (SOA), Application Integration Architecture (AIA), and Repository Creation Utility (RCU).
- Use the sample scripts for secrets and store them in a vault that has strong encryption.
- Secure your Kubernetes secrets by using strong encryption, instead of a default base64 encryption.
- Use Kubernetes Role-Based Access Control (RBAC) on minimum privileges policy and restrict kubectl get, list, and watch privileges for secrets, pods, logs, and services.
- Use Kubernetes RBAC on minimum privileges policy and restrict resource access to pods, such as secrets and network.
- Consider Kubernetes general security guidelines. For details, see the Kubernetes documentation available at: https://kubernetes.io/docs/setup/best-practices/enforcing-pod-security-standards/.

A

Secure Deployment Checklist

The following security checklist contains guidelines to help you secure the Oracle Communications Solution Test Automation Platform (STAP) and its components.

- Configure and deploy only the pre-built integration options you need.
- Enforce strong password management.
- Restrict and control user privileges.
- Restrict network access by doing the following:
  - Use firewalls.
  - Never leave an unnecessary opening in a firewall.
  - Monitor who accesses your systems.
  - Secure the JDBC network connection between the application server and the database physically by using a subnet dedicated to this communication and ensure that the network is not accessible to ordinary users, because network traffic is not encrypted.
  - Install the operating system in a secure location that is difficult for a hacker to access.
  - Apply all security patches and workarounds.
  - Encrypt sensitive information.
  - Contact Oracle support if you discover a vulnerability in any Oracle product.