Hewlett Packard Enterprise

HPE Codar

Software Version: 1.60
Windows ® and Linux operating systems

Configuration Guide

Document Release Date: January 2016
Software Release Date: January 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2015 Hewlett Packard Enterprise Development LP

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: **https://softwaresupport.hp.com**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: **https://hpp12.passport.hp.com/hppcf/createuser.do**

Or click the **the Register** link at the top of the HP Software Support page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HPE Software Support Online web site at: **https://softwaresupport.hp.com**

This web site provides contact information and details about the products, services, and support offered by HPE Software.

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as a Passport user and sign in. Many also require a support contract. To register for a Passport ID, go to:

https://hpp12.passport.hp.com/hppcf/createuser.do

To find more information about access levels, go to:

https://softwaresupport.hp.com/web/softwaresupport/access-levels

HPE Software Solutions Now accesses the HPSW Solution and Integration Portal web site. This site enables you to explore HPE product solutions to meet your business needs, includes a full list of integrations between HPE products, as well as a listing of ITIL processes. The URL for this web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Overview

This document provides information on how to set up the HPE Codar Console and HPE Codar in order to enable users to log in and use the Codar Console . Some tasks must be completed before you can start using Codar.

The user who sets up Codar should have knowledge of or work with someone who has knowledge of LDAP, TLS, HPE Operations Orchestration, and the resource providers that will be integrated with Codar.

Note: If you have added the HPECloud Service Automation license, you have access to all of the Cloud Service Automation functionality, such as global search and reporting database user. For details see the *Cloud Service Automation Configuration Guide*.

The following information is provided in this document:

Getting Started. Before setting up the Codar Console, you may need to complete some initial configuration such as preparing LDAP, configuring Codar truststore properties, and requesting a software license.

Secure Connections. Many of the components that interact with Codar may require communication over a secure connection. You may want to replace the Codar self-signed certificate or configure a secure connection for LDAP, SMTP, the Oracle Database, the Microsoft SQL Server, or the Operations Orchestration Load Balancer.

Operations Orchestration. A process engine whose flows are executed by Codar, Operations Orchestration must be integrated with Codar and sample flows must be imported before the flows can be executed.

The Codar Console. To set up the Codar Console so that users can log in, you must configure the provider organization. In order to start using the Codar Console, you must add a software license. You may wish to import the sample service designs provided with Codar, configure a proxy, or enable or customize tiles in the Codar Console.

Common Codar Tasks. Common tasks include launching the Codar Console, starting, stopping, or restarting Codar, encrypting an Codar password, and uninstalling Codar.

User Administration. User administration includes tasks such as changing the out-of-the-box users. On Windows, also allows non-administrator users to start and stop Codar services.

IPv6 Configuration. Configure Codar to support IPv6 (both dual-stack and IPv6-only).

Common Access Card. Common access cards are used for user authentication and allow users to log in to Codar using a Personal Identity Verification card.

Single Sign-On. Enable or disable Single Sign-On that is included with Codar. Single sign-on can also be configured for the Codar Console with almost any single sign-on solution and a specific solution for CA SiteMinder is provided.

Database Administration. Database administration includes any task that might involve the database, such as configuring the Codar reporting database user if you did not configure it during installation, updating Codar database system or users and passwords, importing large archives, purging service subscriptions, installing the Codar database schema, and configuring Codar to mitigate frequently dropped database connections.

Codar Console Properties. This is a reference to the Codar Console configurable properties.

Operations Orchestration Settings. This is a reference to the Operations Orchestration configurable settings applicable to Codar.

Identity Management Configuration. This is a reference to the Identity Management component configurable settings applicable to Codar.

See the following guides for more information:

- Codar: *Codar Concepts Guide*
- Supported components and versions: *Codar System and Software Support Matrix*
- Installation: *Codar Installation and Configuration Guide*
- Upgrade: *Codar Upgrade Guide*
- Configuration: *Codar Configuration Guide*
- Codar Console: : *Codar Console Help*

Getting started

This chapter provides information about common setup tasks that need to be completed for Codar.

Tasks include:

- "Prepare LDAP for Codar" below (required)
- "Configure Codar truststore properties" on the next page (required)
- "Request software licenses" on page 13 (required)
- "Enable TLS on your web browser" on page 14 (required)
- "Update Codar service startup type" on page 16 (optional)
- "Location of JRE Installed with Codar on Windows" on page 16

Prepare LDAP for Codar

Codar supports limited authentication out-of-the-box and has a fixed set of user names (and associated passwords) that can be used to log in. This basic form of authentication can be used for initial setup and experimentation with the product, but in a production environment, authentication should be configured to occur against a directory service.

Codar can be configured to authenticate against a Lightweight Directory Access Protocol (LDAP) server. Users can then log in with a pre-existing user name (such as an enterprise email address) and password combination. LDAP authenticates the login credentials by verifying that the user name and password match an existing user in the LDAP directory.

In Codar, LDAP is used to:

- Authenticate a user's login to the Codar Console.
- Authenticate a user's access to information.
- Authorize a user's access to information.
- Add user access control functionalities.
- Add users or a group from LDAP to a design for access control.

These functions are configured when you configure LDAP and access control for an organization.

Before you configure LDAP for the Codar Console, you should be familiar with your enterprise LDAP server and LDAP configuration tasks.

> **Note:** The user object configured in LDAP that is used to log in to Codar and by which users can be identified should be configured to contain the following attribute types:
>
> - User Email - Required. This attribute type designates the email address of the user who is to receive email notifications. Common LDAP attribute names for email include **mail**, **email**, and **userPrincipalName**. If the value for this attribute in the user object in LDAP is empty or not valid, the user for whom the value is empty or not valid does not receive email notifications.
> - Group Membership - Required. This attribute type identifies a user as belonging to the group. Common LDAP attribute names that convey group membership include **member** and **uniqueMember**.

The attribute names configured in your LDAP directory for these attribute types are used when configuring an organization's LDAP in the Codar Console

Note: Do not create users in your LDAP directory that match the out-of-the-box users provided by Codar (the out-of-the-box users are `admin`, `csaInboundUser`, `csaCatalogAggregationTransportUser`, `csaReportingUser`, `csaTransportUser`, `idmTransportUser`, `ooInboundUser`, and `codarintegrationUser`). Creating the same users in LDAP may allow the out-of-the-box users unintended access to the Codar Console or give the LDAP users unintended privileges.

Configure Codar truststore properties

You must configure information about the Codar's keystore.

To configure Codar truststore properties, complete the following steps:

1. Open the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file in a text editor.
2. Enter values for the `csaTruststore` and `csaTruststorePassword` properties.

Property	Description
`csaTruststore`	Required. The Codar keystore that stores trusted Certificate Authority certificates. **Note:** Use only forward slashes (/) as your path separators.
`csaTruststorePassword`	Required. The encrypted password of the Codar keystore (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses.

For more information about these properties, see "Codar Console properties" on page 106.

3. Save and exit the file.
4. Restart Codar, see "Restart Codar" on page 60 .

Location of Codar truststore

The location of the Codar truststore depends on the JRE you are using with Codar and where the JRE has been installed.

The following are examples of where the Codar truststore may be located.

- If you are using the JRE that is installed with Codar (OpenJDK JRE), the truststore is located in the following location:

  `CSA_HOME\openjre\lib\security\cacerts`

  `CSA_HOME` is the directory in which Codar is installed — on Windows C:\Program Files\HPE\Codar or on Linux /usr/local/hpe/codar.
- If you are using an Oracle JRE, the truststore may be found in the following location:

`JRE_HOME\lib\security\cacerts`

For example:

Windows: `C:\Program Files\Java\jre7\lib\security\cacerts`

Linux: If you installed the Oracle JRE in `/usr/local/bin`, the truststore may be located at: `/usr/local/bin/jre1.8.0/lib/security/cacerts`

- If you are using an Oracle JDK, the truststore may be found in the following location:

  `JAVA_HOME\lib\security\cacerts`

  For example:

  Windows: `C:\Program Files\Java\jdk1.7.0_71\lib\security\cacerts<JAVA_HOME>/lib/security/cacerts`

  Linux: If you installed the Oracle JDK in `/usr/local/bin`: `/usr/local/bin/jdk1.7.0_71/lib/security/cacerts`

If you still cannot locate the Codar truststore, open the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file and look for the `csaTruststore` property. This property must be set to the location of the Codar truststore after installing Codar.

Request software licenses

Codar version 1.60 requires a software license. Codar licensing is based on the number of operating system instances (OSIs) being used in current, active subscriptions.

After initial installation of Codar version 1.60, when you log in to the Codar Console, a temporary 90-day trial license is activated. Once the trial license expires, you are limited to 25 OSIs. If you created more than 25 OSIs during the trial period, you cannot create any additional OSIs. You can add more licenses at any time to increase your OSI capacity.

After you upgrade to Codar version 1.60, when you log in to the Codar Console, all licenses of earlier versions of Codar are valid and are automatically added.

The following topics are covered in this section:

- "Request software license" below
- "Request software license for clustered environment" on the next page
- "Request software license for system with updated IP address" on the next page

For information on how to view, add, or delete a license, see the *Codar Console Help*.

Request software license

If you received an Electronic Delivery Receipt, use the link to the licensing portal located in the receipt and follow the online instructions to request a software license. Otherwise, to access the licensing portal, go to http://www.hp.com/software/licensing, enter your Entitlement Order Number, and follow the online instructions to request a software license.

See the *Software License Activation Quick Start Guide* for more information about requesting a software license.

IP Address Limitations

When you request a software license, you must supply the IP address (IPv4 or Ipv6) of the system on which Codar is installed.

Do NOT use the following IP addresses when requesting a software license:

- Loopback address - 127.0.0.1 (IPv4) or ::1 (IPv6)

Request software license for clustered environment

If you are configuring Codar in a clustered environment, use the IP address of the load balancer (in the examples given in the *Configuring an Codar Cluster for High Availability Using an Apache Web Server as a Proxy*, this is the APACHE_IP_ADDR; in the examples given in the *Configuring an Codar Cluster for High Availability Using a Load Balancer*, this is the LOAD_BALANCER_IP_ADDR). The license should be installed on only one node in the clustered environment.

Request software license for system with updated IP address

If you change the IP address of the system on which Codar is running, you must request a new software license.

If you immediately add the new license without restarting Codar, the license will not be accepted. You must restart Codar before adding the new license, see "Restart Codar" on page 60. For more information about managing software licenses, see the *Codar Console Help*.

Enable TLS on your web browser

The Codar Console is configured to require https (http over a secure connection) for client browsers. Specifically, the Codar Console is configured to use the TLS protocol. You must enable TLS 1.0 as the required minimum protocol for the browser, and, if applicable, disable the SSL protocols.

Enable your Web browser to use the TLS protocol:

Chrome on Windows:

1. Exit or kill all Chrome sessions.
2. If you added a shortcut to launch Chrome from the Taskbar, remove it: right-click the shortcut on the Taskbar and select **Unpin this program from taskbar**.
3. For every shortcut you use to launch Chrome, do the following:
   a. Right-click on the shortcut and select **Properties**.
   b. Select the **Shortcut** tab.
   c. At the end of the Target field, enter the following after the last quotation mark (and include a space after the last quotation mark but before the following content):

      --ssl-version-min=tls1
   d. Click **OK**.
   e. If asked for administrator privileges, click **Continue**.
4. If you deleted the shortcut from the Taskbar, right-click on any updated shortcut and select **Pin to Taskbar**.
5. If Chrome is your default browser, edit the registry:

a. Click on the **Start** icon, enter **regedit** in the `Search programs and files` box, and press **Enter**.

b. From the Registry Editor, select **HKEY_CLASSES_ROOT > http > shell > open > command**.

c. Double-click **(Default)**.

d. Adding the following at the end of the `Value data` field (and include a space before the following content):

   --ssl-version-min=tls1

e. Click **OK**.

f. Close the Registry Editor dialog.

> **Caution:** Depending on how you launch Chrome, your browser session still may allow SSLv3 connections.

Chrome, Ubuntu

1. Exit or kill all Chrome sessions.
2. Edit the `/usr/share/applications/google-chrome.desktop` file.
3. For every line that starts with `Exec`, add the following argument:

   --ssl-version-min=tls1
4. Save and exit the file.

Chrome, Red Hat Enterprise Linux

1. Exit or kill all Chrome sessions.
2. When invoking the browser from the command line, add the following argument:

   --ssl-version-min=tls1

Microsoft Internet Explorer

1. Open the **Tools** menu (click on the tools icon or type Alt - x) and select **Internet options**.
2. Select the **Advanced** tab.
3. Scroll down to the bottom of the **Settings** section.
4. If TLS is not enabled, select the checkboxes next to **Use TLS 1.0**, **Use TLS 1.1**, and **Use TLS 1.2**.
5. Disable SSL 2.0 and SSL 3.0, if enabled (recommended). Unselect the checkbox next to **Use SSL 2.0** and/or **Use SSL 3.0**.
6. Click **OK**.

Firefox

1. Launch the Firefox browser.
2. In the Location Bar (address bar), enter **about:config** and press **Enter**.
3. In the Search box, enter **security.tls** and press **Enter**.
4. Double-click **security.tls.version.min**.
5. Set the value to **1** and click **OK**.

Update Codar service startup type

If you have services or applications installed on the same system as Codar that Codar requires to be available when Codar is started (such as the database), update the Codar service startup to be delayed. This allows those services time to start before Codar starts if the system is rebooted.

Update Codar service startup type on Windows

To delay the start of the Codar on system reboot, complete the following steps:

1. On the server that hosts Codar, navigate to **Start** > **Administrative Tools** > **Services**.
2. In the Service dialog, right-click on the Codar service and select **Properties**.
3. In the Properties dialog, locate the **Startup type** field and change the value to **Automatic (Delayed Start)**.
4. Click **OK**.

Update Codar service startup type on Linux

To delay the start of the Codar on system reboot, complete the following steps:

```
service codar restart
service codar-execution-service.sh restart
```

Location of JRE Installed with Codar on Windows

The location of the JRE installed with Codar (OpenJDK JRE) is located in the following location:

`CSA_HOME\openjre`

For example: C:\Program Files\HPE\Codar`\openjre`

Secure connections

This chapter provides general information about configuring secure connections between Codar and some commonly used components of Codar. You must consult your security expert for more detailed information about configuring secure connections in your environment.

> **Note:** Codar only accepts secure connections using the TLSv1 protocol. If you are integrating with an application and are using secure connections, you must configure the application to use the TLSv1 protocol with Codar.

Information includes:

- "Configure secure connections for client browsers" below (required when the Codar self-signed certificate expires)
- "Configure secure connections for LDAP" on page 33 (required if the LDAP server requires a secure connection)
- "Configure secure connections for SMTP" on page 33 (required if the SMTP server requires a secure connection)
- "Configure secure connections for Oracle database" on page 34 (required if the Oracle database requires a secure connection)
- "Configure secure connections for Microsoft SQL server" on page 36 (required if Microsoft SQL Server requires a secure connection)
- "Configure secure connections for Operations Orchestration Load Balancer" on page 37 (required if you are running the Operations Orchestration Load Balancer server and it requires a secure connection)

The function of the secure connection is configured by the `com.hp.csa.service.ssl.insecure` property in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file. That is, a secure connection can be configured to only authenticate the certificate root and verify that the certificate presented by another application or component has not been revoked (default). Or, a secure connection can be configured to authenticate the certificate root, verify that the certificate presented by another application or component has not been revoked, verify the certificate's validity (beginning and expiration dates), and validate the certificate's hostname (configured as the certificate's common name). See the *Secure Connections* section in "Codar Console properties" on page 106 for more information about the `com.hp.csa.service.ssl.insecure` property.

Configure secure connections for client browsers

The Codar Console is configured to require https (http over a secure connection) for client browsers. For a secure connection to be established, a certificate must first be installed on theCodar server.

A self-signed certificate is created and configured whenCodar is installed and is configured with the fully-qualified domain name that was entered during the installation. This self-signed certificate is used when https browser requests are issued for the Codar Console and expires 120 days after Codar is installed.

When client browsers connect to the Codar Console in this default configuration, the client browser will usually issue warnings that the certificate was not issued by a trusted authority. The end user can choose to continue to the web site or close the browser.

Although the self-signed certificate can be used in production, HPE recommends that you replace this certificate by configuring a trusted third-party Certificate Authority-signed or subordinate Certificate Authority-signed certificate (see "Configure Codar to use trusted Certificate Authority-Signed or subordinate Certificate Authority-Signed certificate" below) or by configuring an internal Certificate Authority-signed certificate (see "Configure Codar to use internal Certificate Authority-Signed certificate" on page 22). Or, you can replace this certificate by configuring a self-signed certificate (see "Configure Codar to use self-signed certificate" on page 25).

> **Note:**Certificate chains require additional configuration and general information about importing a chain of certificates is provided in this section. However, you should consult your security expert for more detailed information when using certificate chains in your environment. Wildcard certificates do not require special configuration.

Configure Codar to use trusted Certificate Authority-Signed or subordinate Certificate Authority-Signed certificate

This section describes the process you should follow to obtain, install, and configure a trusted third-party Certificate Authority-signed or subordinate Certificate Authority-signed certificate for use by Codar. The process by which you acquire a certificate depends on your organization. If you are obtaining a certificate from a trusted third-party Certificate Authority, such as Verisign, perform the following general steps, which are described in detail below. If you are generating and/or obtaining a certificate from an internal Certificate Authority, such as a corporate Certificate Authority, you should perform the general steps in "Configure Codar to use internal Certificate Authority-Signed certificate" on page 22.

"Step 1: Create a keystore and self-signed certificate" below

"Step 2: Create a Certificate Signing Request" on the next page

"Step 3: Submit the certificate signing request to a Certificate Authority" on the next page

"Step 4: Import the Certificate Authority's root certificate" on page 20

"Step 5: Import Certificate Authority-Signed certificate" on page 20

"Step 6: Configure the web server" on page 21

"Step 7: Configure client browsers" on page 22

"Step 8: Test secure connections" on page 22

> **Note:** In the following instructions, `CSA_HOME` is the directory in which Codar is installed (for example, on Windows, the directory is C:\Program Files\HPE\Codar and on Linux, the directory is /usr/local/hpe/codar). The `keytool` utility is included with the JRE.
>
> Also, the following instructions are applicable for subordinate Certificate Authorities. Wherever the Certificate Authority is mentioned, the subordinate Certificate Authority is implied. For example, if the content states to submit the certificate to a Certificate Authority, you may also submit the certificate to a subordinate Certificate Authority.

Step 1: Create a keystore and self-signed certificate

Create a self-signed certificate to send with your request to a Certificate Authority by completing the following steps:

1. Open a command prompt and change directories to `CSA_HOME`.
2. Run the following command:

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -genkeypair -alias codar_ca_signed
   -validity 365 -keyalg rsa -keysize 2048 -keystore
   .\jboss-as\standalone\configuration\.keystore_ca_signed
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/keytool -genkeypair -alias codar_ca_signed-validity 365 -keyalg rsa
   -keysize 2048 -keystore./jboss-as/standalone/configuration/.keystore_ca_signed
   ```

   `CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed.

   You can use different values for `-alias`, `-validity`, `-keysize` and `-keystore`. These instructions assume that you will use the `-alias` and `-keystore` values recommended here. You will need to adjust the commands accordingly if you use different values.
3. Enter a keystore password.

   This password is used to control access to the keystore. This password must be the same as the password you enter for the key later in this procedure.
4. When you are prompted for your first and last name, enter the fully qualified domain name of the Codar server.
5. Follow the prompts to enter the remaining organization and location values.
6. Enter the keystore password you supplied earlier to use as the key password.

   Although `keytool` allows you to enter different passwords for the keystore and the key, the two passwords must be the same to work with Codar.

Step 2: Create a Certificate Signing Request

To enable a Certificate Authority to sign the self-signed certificate, you will need to create a Certificate Signing Request using the following procedure:

1. Open a command prompt and change directories to `CSA_HOME`.
2. Run the following command:

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -certreq -alias codar_ca_signed
   -file C:\codarcsr.txt -keystore .\jboss-as\standalone\configuration\.keystore_ca_
   signed
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/keytool -certreq -alias codar_ca_signed-file /tmp/codarcsr.txt -
   keystore ./jboss-as/standalone/configuration/.keystore_ca_signed
   ```
3. When you are prompted for a password, enter the password you supplied for the keystore and key when you created the keystore and self-signed certificate in step 1.

Step 3: Submit the certificate signing request to a Certificate Authority

Submit the Certificate Signing Request to the Certified Authority following the procedure used by your organization or the third-party provider. After the submission has been processed, you will receive a Certificate Authority-signed certificate and a root certificate for the Certificate Authority.

In this example, it is assumed that the Certificate Authority's root certificate is named `codarca.cer`, the Certificate Authority-signed certificate is named `codar_ca_signed.cer`, and that both are located in `C:\` on Windows or `/tmp` on Linux.

Step 4: Import the Certificate Authority's root certificate

This step configures the JRE so it trusts the Certificate Authority that has signed your certificate. The JRE ships with a list of common, trusted Certificate Authority certificates that are stored in a keystore named `cacerts`. If the Certificate Authority used to sign your certificate is well known, it is likely that this root certificate is already present in the `cacerts` keystore. It is recommended that you perform the following steps even if you suspect that the certificate is already installed. The `keytool` command will detect if the certificate is already present, and you can exit the import process if the certificate exists.

1. Open a command prompt.
2. Run the following command:

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -alias codarca -file C:\codarca.cer -
   trustcacerts -keystore "CSA_JRE_HOME\lib\security\cacerts"
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/keytool -importcert -alias codarca -file /tmp/codarca.cer -
   trustcacerts -keystore CSA_JRE_HOME/lib/security/cacerts
   ```
3. When prompted enter the password for the keystore.
4. Enter `yes` when prompted to trust the certificate.

Step 5: Import Certificate Authority-Signed certificate

1. The Certificate Authority-signed certificate (`codar_ca_signed.cer`) contains a chain of certificates and you must copy the root and any intermediate certificates in the chain to separate files. Work with your security expert to copy each certificate to a separate file.
2. Open a command prompt and change directories to `CSA_HOME`.
3. Import the certificate file(s). Import each separate file in the following order (each certificate must have a unique alias):
   - root certificate
   - intermediate or subordinate certificate(s) in hierarchical order

     primary or end-user certificate

     For example, if the Certificate Authority-signed certificate contains three certificates (root, intermediate, and primary) and you copied the root certificate to `/tmp/root.cer` and the intermediate certificate to `/tmp/intermediate.cer` (you will use the Certificate Authority-signed certificate as the primary certificate), run the following commands in the following order to import each certificate:

     Windows:

     ```
     "CSA_JRE_HOME\bin\keytool" -importcert -alias codar_ca_signed
     -file C:\codar_ca_signed.cer -trustcacerts -keystore
     .\jboss-as\standalone\configuration\.keystore_ca_signed
     ```

     Linux:

```
CSA_JRE_HOME/bin/keytool -importcert -alias codar_ca_signed
-file /tmp/codar_ca_signed.cer -trustcacerts -keystore
./jboss-as/standalone/configuration/.keystore_ca_signed
```

Use the alias of the primary certificate (`codar_ca_signed`) and keystore name when you configure the Web server.

4. When prompted, enter the password for the key and keystore.

   Use this password when you configure the Web server.

Step 6: Configure the web server

Configure the web server by completing the following steps:

1. Open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor.
2. Locate the following entry:

   ```
   <keystore path="CSA_HOME/jboss-as/standalone/configuration/.keystore"
   keystore-password="changeit"/>
   ```

3. Set the `path` attribute to the keystore you used in step 2, set the `keystore-password` attribute to the value that corresponds to the password you selected for the keystore, and add the `key-alias` attribute and set it to the alias you used in step 2.

   ```
   <keystore path="CSA_HOME/jboss-as/standalone/
   configuration/.keystore_self_signed" keystore-password="keystorePassword"
   alias="csa_self_signed"/>
   ```

   Note: This example stores the password in clear text. If you want to use an encrypted password, follow the instructions at https://community.jboss.org/wiki/JBossAS7SecuringPasswords to create a password vault for JBoss.

   Note: If you are using the vault scripts, verify that the `JAVA_HOME` environment variable has been defined. Verify that `JAVA_HOME` has been set to the directory in which the JRE that is used by Codar is installed.

   Windows:

   If the directory path name includes a space, verify that the value has been enclosed in quotations marks. For example, to set `JAVA_HOME` to a directory path name that includes a space, from a command prompt, type

   ```
   set JAVA_HOME="C:\Program Files\HPE\Codar\jre"
   ```

   To verify that `JAVA_HOME` has been defined, from a command prompt, type:

   ```
   echo %JAVA_HOME%
   ```

   Linux:

   To verify that `JAVA_HOME` has been defined, from a command prompt, type:

   ```
   echo $JAVA_HOME
   ```

   The following is an example of an encrypted password attribute using the JBoss password vault:

   ```
   password="${VAULT::<vault_block_example>::password::N2NhZDzOMtES0ZGE4MmEtx0}"
   ```

4. Restart Codar service, see "Restart Codar" on page 60.

5. After the service has started, review the log files in `CSA_HOME\jboss-as\standalone\log\` and verify that no TLS or keystore errors are present.

Step 7: Configure client browsers

The client browser must be configured to trust certificates that are signed by the Certificate Authority. In most situations, this step will already have occurred. Client browsers are likely to already trust well-known third-party Certificate Authorities, or will have previously accessed and trusted Web sites that use internal Certificate Authority root certificates.

To test whether or not the browser on a client system is configured to trust certificates signed by your Certificate Authority, open a supported Web browser and navigate to `https://<codarhostname>:8444/csa`. If you do not see a certificate warning, then the browser is configured properly.

If client browsers need to be configured to trust certificates signed by your Certificate Authority, then you will need to make the root certificate available to clients so it can be installed in the browser. The process of installing the root certificate will vary based on the browser.

- **Microsoft Internet Explorer** and **Chrome**: From Windows Explorer, double-click on the `.cer` file to begin the import process. Install the certificate in the Trusted Root Certification Authorities store. For information on how to import the certificate, see the browser's online documentation.
- **Firefox**: To begin the import process, select **Tools > Options**, select **Advanced**, select the **Encryption** tab, and click **View Certificates**. Import the root certificate into the Authorities tab. For information on how to import the certificate, see the browser's online documentation.

Step 8: Test secure connections

To test the connection to the Codar Console, on a client system, open a supported Web browser and navigate to `https://`*`<codarhostname>`*`:8444/csa` where *`<codarhostname>`* is the fully-qualified domain name of the system that was used when the certificate was created. If the client browser is configured to accept the Certificate Authority's root certificate and the web application opens without a certificate warning, then you have successfully configured Codar to use a Certificate Authority-signed certificate. If a certificate warning is displayed, review steps 1-7 to be sure they were followed as documented.

Configure Codar to use internal Certificate Authority-Signed certificate

This section describes the process you should follow to install and configure an internal root and internal Certificate Authority-signed certificate for use by Codar. An internal certificate is one that is generated by an internal Certificate Authority, such as a corporate or government Certificate Authority. For an internal Certificate Authority, you do not have to generate a self-signed certificate nor create a certificate signing request. The internal Certificate Authority should provide you with a root certificate and signed certificate.

Perform the following general steps:

1. "Step 1: Import the Certificate Authority's root certificate" on the next page
2. "Step 2: Import internal Certificate Authority-Signed certificate" on the next page
3. "Step 3: Configure the web server" on the next page
4. "Step 4: Configure client browsers" on page 25
5. "Step 5: Test secure connections" on page 25

Note: In the following instructions, `CSA_HOME` is the directory in which Codar is installed (for example, on Windows the directory is C:\Program Files\HPE\Codar and on Linux the directory is `/usr/local/hpe/codar`). The `keytool` utility is included with the JRE.

In this example, it is assumed that you are given an internal Certificate Authority-signed certificate (referred to as `codar_internalca_signed.cer`), an internal Certificate Authority's root certificate (referred to as `codarinternalca.cer`). Both certificates are located in C:\ on Windows or in /tmp on Linux.

Step 1: Import the Certificate Authority's root certificate

This step configures the JRE so it trusts the internal Certificate Authority that has signed your certificate by importing the internal Certificate Authority into a keystore named `cacerts` that is shipped with the JRE.

1. Open a command prompt.
2. Run the following command:

   On Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -alias codarinternalca -file
   C:\codarinternalca.cer -trustcacerts -keystore "CSA_JRE_HOME\lib\security\cacerts"
   ```

   On Linux:

   ```
   CSA_JRE_HOME/bin/keytool -importcert -alias codarinternalca -
   file/tmp/codarnternalca.cer -trustcacerts -keystore CSA_JRE_
   HOME/lib/security/cacerts
   ```

   `CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed.
3. When prompted enter the password for the keystore.
4. Enter **yes** when prompted to trust the certificate.

Step 2: Import internal Certificate Authority-Signed certificate

1. Open a command prompt and change directories to `CSA_HOME`.
2. Run the following command:

   On Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -alias codar_internalca_signed
   -file C:\codar_internalca_signed.cer -trustcacerts -keystore
   .\jboss-as\standalone\configuration\.keystore_internalca_signed
   ```

   On Linux:

   ```
   CSA_JRE_HOME/bin/keytool -importcert -alias codar_internalca_signed
   -file /tmp/codar_internalca_signed.cer -trustcacerts -keystore
   ./jboss-as/standalone/configuration/.keystore_internalca_signed
   ```

   `CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed.

   Use this alias and keystore name when you configure the web server.
3. When prompted, enter the password for the key and keystore.

   Use this password when you configure the web server.

Step 3: Configure the web server

Configure the web server by completing the following steps:

1. Open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor.
2. Locate the following entry:

   ```
   <ssl name="ssl" key-alias="CODAR" certificate-key-file=
   "CSA_HOME\jboss-as\standalone\configuration\
   .keystore verify-client="false"/>
   ```

3. Add a new attribute named `password` with a value that corresponds to the password you selected for the keystore, change the name of the `key-alias` to the alias you used in step 2, and change the name of the `certificate-key-file` to the keystore you used in step 2.

   On Windows:

   ```
   <ssl name="ssl" key-alias="codar_self_signed" certificate-key-file="
   "CSA_HOME\jboss-as\standalone\configuration\
   .keystore_internalca_signed" password="keystorePassword"
   verify-client="false"/>
   ```

   On Linux:

   ```
   <ssl name="ssl" key-alias="codar_ca_signed" certificate-key-file=
   CSA_HOME/jboss-as/standalone/configuration/
   .keystore_ca_signed" password="keystorePassword"
   verify-client="false"/>
   ```

   Note: This example stores the password in clear text. If you want to use an encrypted password, follow the instructions at https://community.jboss.org/wiki/JBossAS7SecuringPasswords to create a password vault for JBoss.

   Note: If you are using the vault scripts, verify that the `JAVA_HOME` environment variable has been defined. Verify that `JAVA_HOME` has been set to the directory in which the JRE that is used by Codar is installed.

   Windows:

   If the directory path name includes a space, verify that the value has been enclosed in quotations marks. For example, to set `JAVA_HOME` to a directory path name that includes a space, from a command prompt, type

   ```
   set JAVA_HOME="C:\Program Files\HPE\Codar\jre"
   ```

   To verify that `JAVA_HOME` has been defined, from a command prompt, type:

   ```
   echo %JAVA_HOME%
   ```

   Linux:

   To verify that `JAVA_HOME` has been defined, from a command prompt, type:

   ```
   echo $JAVA_HOME
   ```

   The following is an example of an encrypted password attribute using the JBoss password vault:`password="${VAULT::<vault_block_example>::password::N2NhZDzOMtES0ZGE4MmEtx0}"`

4. Restart the Codar service, see "Restart Codar" on page 60.
5. After the service has started, review the log files in `CSA_HOME\jboss-as\standalone\log\` and verify that no TLS or keystore errors are present.

Step 4: Configure client browsers

The client browser must be configured to trust certificates that are signed by the Certificate Authority. In most situations, this step will already have occurred. Client browsers are likely to already trust well-known third-party Certificate Authorities, or will have previously accessed and trusted Web sites that use internal Certificate Authority root certificates.

To test whether or not the browser on a client system is configured to trust certificates signed by your Certificate Authority, open a supported Web browser and navigate to `https://<codarhostname>:8444/csa`. If you do not see a certificate warning, then the browser is configured properly.

If client browsers need to be configured to trust certificates signed by your Certificate Authority, then you will need to make the root certificate available to clients so it can be installed in the browser. The process of installing the root certificate will vary based on the browser.

- **Microsoft Internet Explorer** and **Chrome**: From Windows Explorer, double-click on the `.cer` file to begin the import process. Install the certificate in the Trusted Root Certification Authorities store. For information on how to import the certificate, see the browser's online documentation.
- **Firefox**: To begin the import process, select **Tools > Options**, select **Advanced**, select the **Encryption** tab, and click **View Certificates**. Import the root certificate into the Authorities tab. For information on how to import the certificate, see the browser's online documentation.

Step 5: Test secure connections

To test the connection to the Codar Console, on a client system, open a supported Web browser and navigate to `https://`*`<codarhostname>`*`:8444/csa` where *`<codarhostname>`* is the fully-qualified domain name of the system that was used when the certificate was created. If the client browser is configured to accept the Certificate Authority's root certificate and the web application opens without a certificate warning, then you have successfully configured Codar to use a Certificate Authority-signed certificate. If a certificate warning is displayed, review steps 1-4 to be sure they were followed as documented.

Configure Codar to use self-signed certificate

This section describes the process you should follow to obtain, install, and configure a self-signed certificate for use by Codar.

In general, recommends that you replace Codar's self-signed certificate with a Certificate Authority-signed certificate. However, you may consider replacing Codar's self-signed with a self-signed certificate you create in the following situations:

- Codar's self-signed certificate has expired and you do not want to configure a Certificate Authority-signed certificate at this time.
- You want to configure a certificate with a hostname that matches the Codar hostname to avoid certain browser warnings that occur when accessing the Codar Console.
- The hostname that you entered when you installed Codar has changed (the hostname you entered during installation is used to configure Codar's self-signed certificate).
- You entered an IP address instead of the fully-qualified domain name when Codar was installed.
- Obtaining a Certificate Authority-signed certificate is not an option in your environment.

You should perform the following general steps:

1. "Step 1: Create a keystore and self-signed certificate" below
2. "Step 2: Export the self-signed certificate" on the next page
3. "Step 3: Import self-signed certificate as a trusted certificate" on the next page
4. "Step 4: Configure web server" on the next page
5. "Step 5: Configure client browsers (optional)" on page 28
6. "Step 6: Test secure connections" on page 28

Note: In the following instructions, `CSA_HOME` is the directory in which Codar is installed (for example, on Windows, the directory is C:\Program Files\HPE\Codar) and on Linux the directory is `/usr/local/hpe/codar`). The `keytool` utility is included with the JRE.

Step 1: Create a keystore and self-signed certificate

Create a self-signed certificate by completing the following steps:

1. Open a command prompt and change directories to `CSA_HOME`.
2. Run the following command:

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -genkeypair -alias codar_self_signed
   -validity 365 -keyalg rsa -keysize 2048
   -keystore .\jboss-as\standalone\configuration\
   .keystore_self_signed [-ext san=ip:<ip_address>]
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/keytool -genkeypair -alias codar_self_signed
   -validity 365 -keyalg rsa -keysize 2048
   -keystore./jboss-as/standalone/configuration/
   .keystore_self_signed [-ext san=ip:<ip_address>]
   ```

   `CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed and `-ext san=ip:<ip_address>` is the option to specify the IP address of the system on which Codar is installed. This option is required if you specified an IP address instead of the fully qualified domain name when you installed Codar. If you specified the fully-qualified domain name during installation, you may omit this option.

   You can use different values for `-alias`, `-validity`, `-keysize` and `-keystore`. These instructions assume that you will use the `-alias` and `-keystore` values recommended here; you will have to adjust the commands accordingly if you use different values.
3. Enter a keystore password.

   This password is used to control access to the keystore. This password must be the same as the password you enter for the key later in this procedure.
4. When you are prompted for your first and last name, enter the fully qualified domain name of the Codar server.
5. Follow the prompts to enter the remaining organization and location values.
6. Enter the keystore password you supplied earlier to use as the key password.

   Although `keytool` allows you to enter different passwords for the keystore and the key, the two passwords must be the same to work with Codar.

Step 2: Export the self-signed certificate

Export the self-signed certificate completing the following steps:

1. Open a command prompt and change directories to `CSA_HOME`.
2. Run the following command:

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -export -alias codar_self_signed
   -file C:\codar_self_signed.cer
   -keystore .\jboss-as\standalone\configuration\
   .keystore_self_signed
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/keytool -export -alias codar_self_signed
   -file /tmp/codar_self_signed.cer
   -keystore ./jboss-as/standalone/configuration/
   .keystore_self_signed
   ```

   `CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed
3. When you are prompted for a password, enter the keystore password used in step 1.

Step 3: Import self-signed certificate as a trusted certificate

This step configures the JRE to trust the self-signed certificate. Import the self-signed certificate by completing the following steps:

1. Open a command prompt.
2. Run the following command:

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -alias codar_self_signed
   -file C:\codar_self_signed.cer -trustcacerts
   -keystore "CSA_JRE_HOME\lib\security\cacerts"
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/keytool -importcert -alias codar_self_signed
   -file /tmp/codar_self_signed.cer -trustcacerts
   -keystore CSA_JRE_HOME/lib/security/cacerts
   ```

   `CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed.
3. When you are prompted for a password, enter the keystore password used in step 1.
4. Enter `yes` when prompted to trust the certificate.

Step 4: Configure web server

Configure the web server by completing the following steps:

1. Open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor.
2. Locate the following entry:

```
<keystore path=
"CSA_HOME\jboss-as\standalone\configuration\
.keystore" keystore-password="changeit"/>
```

3. Set the `path` attribute to the keystore you used in step 2, set the `keystore-password` attribute to the value that corresponds to the password you selected for the keystore, and add the `key-alias` attribute and set it to the alias you used in step 2.

   Windows:

   ```
   <keystore path="<CSA_HOME>\jboss-as\standalone\configuration\.keystore_self_signed"
   keystore-password="keystorePassword"
   alias="csa_self_signed"/>
   ```

   Linux:

   ```
   <keystore path="$CSA_HOME/jboss-as/standalone/
   configuration/.keystore_self_signed" keystore-password="keystorePassword"
   alias="csa_self_signed"/>
   ```

   Note: This example stores the password in clear text. If you want to use an encrypted password, follow the instructions at https://community.jboss.org/wiki/JBossAS7SecuringPasswords to create a password vault for JBoss.

3. Restart the Codar service, see "Restart Codar" on page 60.
4. After the service has started, review the log files in `CSA_HOME\jboss-as\standalone\log\` and verify that no TLS or keystore errors are present.

Step 5: Configure client browsers (optional)

Because the self-signed certificate is not signed by a Certificate Authority, when accessing the Codar Console, warning messages are displayed in the browser (these messages do not affect normal operations of Codar). To avoid these warning messages, import the `codar_self_signed.cer` file or add an exception.

- **Microsoft Internet Explorer** and **Chrome**: From Windows Explorer, double-click on the `codar_self_signed.cer` file to begin the import process. Install the certificate in the Trusted Root Certification Authorities store. For information on how to import the certificate, see the browser's online documentation.
- **Firefox**: Add an exception by opening the browser and navigating to `https://<codarhostname>:8444/csa` where *<codarhostname>* is the fully-qualified domain name of the system on which Codar is running. When the **This Connection is Untrusted** page opens, select **I Understand the Risks**, click the **Add Exception** button, verify the Server Location, and click **Confirm Security Exception**. For information on how to import the certificate, see the browser's online documentation.

Step 6: Test secure connections

To test the connection to the Codar Console, on a client system, open a supported Web browser and navigate to `https://<codarhostname>:8444/csa` where *<codarhostname>* is the fully-qualified domain name of the system that was used when the certificate was created. If the client browser is configured to accept the Certificate Authority's root certificate and the web application opens without a certificate warning, then you have successfully configured Codar to use a Certificate Authority-signed certificate. If any other certificate warning is displayed, review steps 1-5 to be sure they were followed as documented.

Masking Passwords in standalone.xml Using the JBoss vault Script

JBoss provides a script that allows passwords in the `standalone.xml` file to be masked. The following tasks describe how to use the JBoss vault script and configure [[[Undefined variable CSAVariables.CSA long name]]] to use the masked password.

1. Verify that the `JAVA_HOME` environment variable has been defined and that `JAVA_HOME` has been set to the directory in which the JRE that is used by [[[Undefined variable CSAVariables.CSA long name]]] is installed (for example, on Windows: `C:\Program Files\HPE\CSA\openjre` and on Linux: `/usr/local/hpe/csa/openjre`).

   > **Note:** Do NOT enclose the value in quotation marks, even if the path name includes a space. The vault script will fail if the JAVA_HOME variable definition contains quotation marks.

   To verify that `JAVA_HOME` has been defined, from a command prompt, type:

   ```
   echo JAVA_HOME
   ```

2. Create a keystore used by vault. This vault keystore is used to store the [[[Undefined variable CSAVariables.CSA long name]]] keystore password.

   > **Note:** This example saves the vault keystore and encrypted vault file in the `CSA_HOME\[[[Undefined variable CSAVariables.dirjboss]]]\standalone\configuration\` directory (the contents of this directory are automatically backed up during an upgrade). You may choose to store the vault keystore and encrypted vault file in any location. However, you must remember to use those locations in subsequent steps in this task and, if those locations are not automatically backed up during upgrade, to manually back up the files before upgrade.

   a. Open a command prompt.

   b. Run the following command:

      Windows:

      ```
      "CSA_JRE_HOME\bin\keytool" -genkey -alias vault -validity 365 -keyalg rsa
      -keysize 2048 -keystore .\[[[Undefined variable CSAVariables.dirjboss]]]
      \standalone\configuration\csa_vault.keystore
      ```

      Linux:

      ```
      CSA_JRE_HOME/bin/keytool -genkey -alias vault -validity 365 -keyalg rsa
      -keysize 2048 -keystore ./[[[Undefined variable CSAVariables.dirjboss]]]
      /standalone/configuration/csa_vault.keystore
      ```

      where

      `CSA_JRE_HOME` is the directory in which the JRE that is used by [[[Undefined variable CSAVariables.CSA long name]]] is installed

      You can use different values for `-alias`, `-validity`, `-keysize` and `-keystore`. These instructions assume that you will use the `-alias` and `-keystore` values recommended here; you will have to adjust the commands accordingly if you use different values.

   c. Enter the vault keystore password (for example, csavault).

      This password is used to control access to the vault keystore. This password must be the same as the password you enter for the key in step e of this task.

   d. Follow the prompts to enter your first and last name, organization, and location values.

e. Enter the key password. Click **Enter** to use the vault keystore password you supplied earlier (for example, csavault).

   Although `keytool` allows you to enter different passwords for the keystore and the key, the two passwords must be the same to work with [[[Undefined variable CSAVariables.CSA long name]]].

3. Run the vault script. The script will generate the masked password and the values to configure in the `standalone.xml` file in order to use the masked password.

   a. On Linux from the command prompt, make the vault script executable. Type: `chmod 775 CSA_HOME/[[[Undefined variable CSAVariables.dirjboss]]]/bin/vault.sh`

   b. From the command prompt, type:

      Windows:

      ```
      CSA_HOME\[[[Undefined variable CSAVariables.dirjboss]]]\bin\vault
      ```

      Linux:

      ```
      CSA_HOME/[[[Undefined variable CSAVariables.dirjboss]]]/bin/vault.sh
      ```

   c. Select **0** to start the interactive session.

   d. Enter the following information, when prompted, to configure the vault keystore:

Prompt	Description
Directory to store encrypted files	Directory in which the vault encrypted file is stored (for example, `CSA_HOME\[[[Undefined variable CSAVariables.dirjboss]]]\standalone\configuration`). Verify that a vault encrypted file (`VAULT.dat` on Windows or `ENC.dat` on Linux) does not already exist in this directory. If the file exists, select a different directory.
Keystore URL	The name and location of the vault keystore (for example, `CSA_HOME\[[[Undefined variable CSAVariables.dirjboss]]]\standalone\configuration\csa_vault.keystore`).
Keystore password (twice)	The password to the vault keystore (for example, csavault).
8 character salt	A random number (for example, 12345678).
Iteration count as a number	The number of times the [[[Undefined variable CSAVariables.CSA long name]]] keystore password is hashed (for example, 25).
Keystore alias	The alias used to identify the [[[Undefined variable CSAVariables.CSA long name]]] keystore password in the vault keystore (for example, vault).

   e. Make a copy of the vault property block that is displayed. For example, copy:

```
<vault>
   <vault-option name="KEYSTORE_URL" value="CSA_HOME\[[[Undefined variable
CSAVariables.dirjboss]]]\standalone\configuration\csa_vault.keystore"/>
   <vault-option name="KEYSTORE_PASSWORD" value="MASK-2PtpNyQsI1E7t"/>
   <vault-option name="KEYSTORE_ALIAS" value="vault"/>
```

```
   <vault-option name="SALT" value="12345678"/>
   <vault-option name="ITERATION_COUNT" value="25"/>
   <vault-option name="ENC_FILE_DIR" value="CSA_HOME\[[[Undefined variable
CSAVariables.dirjboss]]]\standalone\configuration\"/>
</vault>
```

You will need to add this content to the `standalone.xml` file (the exact location is described in a later step).

f. Select **0** to store a secured attribute.

g. Enter the following information, when prompted, to generate the vault entry to use for the [[[Undefined variable CSAVariables.CSA long name]]] keystore password in the `standalone.xml` file:

Prompt	Description
Secured attribute value (twice)	Enter the [[[Undefined variable CSAVariables.CSA long name]]] keystore password (for example, changeit).
Vault Block	Enter a name for the vault block (for example, csa_keystore).
Attribute Name	Enter the attribute being stored (for example, password).

Note the VAULT entry (for example, `VAULT::csa_keystore::password::1`). You will need this value when you configure the `standalone.xml` file.

h. Enter **2** to exit the script.

> **Note:** The vault script converts the format of the vault keystore (for example, CSA_HOME\[[[Undefined variable CSAVariables.dirjboss]]]\standalone\configuration\csa_vault.keystore) to JCEKS.

4. Open `CSA_HOME\[[[Undefined variable CSAVariables.dirjboss]]]\standalone\configuration\standalone.xml` in a text editor.

5. Locate the following entry for the [[[Undefined variable CSAVariables.CSA long name]]] server keystore (this entry may have been modified):

```
<keystore path="CSA_HOME/[[[Undefined variable CSAVariables.dirjboss]]]
/standalone/configuration/.keystore" keystore-password="changeit"/>
```

6. Update the entry by changing the value of the `keystore-password` attribute to the vault entry you generated (for example, `VAULT::csa_keystore::password::1`).

   For example:

```
<keystore path="CSA_HOME/[[[Undefined variable CSAVariables.dirjboss]]]
/standalone/configuration/.keystore" keystore-password="${VAULT::csa_
keystore::password::1}"/>
```

Add the vault property block to `<server xmlns="urn:jboss:domain:1.3">` after the `system-properties` block. For example, using the example values, enter the following:

```
<server xmlns="urn:jboss:domain:1.3">
 .
 .
 .
<system-properties>
 .
 .
 .
</system-properties>
<vault>
   <vault-option name="KEYSTORE_URL" value="CSA_HOME\[[[Undefined variable
CSAVariables.dirjboss]]]\standalone\configuration\csa_vault.keystore"/>
   <vault-option name="KEYSTORE_PASSWORD" value="MASK-2PtpNyQsI1E7t"/>
   <vault-option name="KEYSTORE_ALIAS" value="vault"/>
   <vault-option name="SALT" value="12345678"/>
   <vault-option name="ITERATION_COUNT" value="25"/>
   <vault-option name="ENC_FILE_DIR" value="CSA_HOME\[[[Undefined variable
CSAVariables.dirjboss]]]\standalone\configuration\"/>
</vault>
```

Note: In a clustered environment, add the `vault` xml entries in `host.xml` as shown below.

For example, using the example value, enter the following:

```
Host.xml -
<?xml version='1.0' encoding='UTF-8'?>
<host name="master_node" xmlns="urn:jboss:domain:1.2">

<vault>
<vault-option name="KEYSTORE_URL" value="%CSA_
HOME%\jbossas\standalone\configuration\csa_vault.keystore"/>
<vault-option name="KEYSTORE_PASSWORD" value="MASK-2PtpNyQsI1E7t"/>
<vault-option name="KEYSTORE_ALIAS" value="vault"/>
<vault-option name="SALT" value="12345678"/>
<vault-option name="ITERATION_COUNT" value="25"/>
<vault-option name="ENC_FILE_DIR" value="%CSA_HOME%\jbossas\
standalone\configuration\"/>
</vault>

<management>
        <security-realms>
           <security-realm name="ManagementRealm">
              <authentication>
                 <properties path="mgmt-users.properties" relative-
to="jboss.domain.config.dir"/>
              </authentication>
           </security-realm>
           <security-realm name="ApplicationRealm">
              <authentication>
                 <properties path="application-users.properties" relative-
to="jboss.domain.config.dir" />
              </authentication>
           </security-realm>
        </security-realms>
        <management-interfaces>
           <native-interface security-realm="ManagementRealm">
              <socket interface="management"
port="${jboss.management.native.port:9999}"/>
           </native-interface>
           <http-interface security-realm="ManagementRealm">
```

```
              <socket interface="management" port="${jboss.management.http.port:9990}
"/>
           </http-interface>
        </management-interfaces>
</management>
```

Configure secure connections for LDAP

If the LDAP server requires a secure connection, follow these steps to import the LDAP server Certificate Authority's root certificate into the Java truststore of Codar. If necessary, contact your LDAP administrator to obtain the LDAP server certificate.

If the LDAP server does not require a secure connection, you can omit this task.

1. Open a command prompt and run the `keytool` utility with the following options to create a local trusted certificate entry for the LDAP server.

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -trustcacerts -alias ldap
   -keystore "CSA_JRE_HOME\lib\security\cacerts"
   -file <c:\certfile_name.cer> -storepass <password>
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/keytool -importcert -trustcacerts -alias ldap
   -keystore CSA_JRE_HOME/lib/security/cacerts
   -file </tmpcertfile_name.cert> -storepass <password>
   ```

   `<c:\certfile_name.cer>` on Windows or `</tmp/certfile_name.cer>` on Linux is the path and name of the Certificate Authority's root certificate for the LDAP server. The file extension may be `.cer` rather than `.crt`. You can also use a different value for `-alias`.
2. At the prompt to import the certificate, type **Yes**.
3. Press **Enter**.
4. Restart Codar service, see "Restart Codar" on page 60.

Configure secure connections for SMTP

For each organization, if its SMTP server requires a secure connection, follow these steps to import the SMTP server Certificate Authority's root certificate into the Java truststore of Codar. If necessary, contact your SMTP server administrator to obtain the SMTP server certificate.

If the SMTP server does not require a secure connection, you can omit this task.

1. Open a command prompt and run the `keytool` utility with the following options to create a local trusted certificate entry for the SMTP server.

   Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -trustcacerts -alias smtp
   -keystore "CSA_JRE_HOME\lib\security\cacerts"
   -file <c:\certfile_name.cer> -storepass <password>
   ```

   Linux:

```
CSA_JRE_HOME/bin/keytool -importcert -trustcacerts -alias smtp
-keystore CSA_JRE_HOME/lib/security/cacerts
-file </tmp/certfile_name.cer> -storepass <password>
```

`<c:\certfile_name.cer>` on Windows or `</tmp/certfile_name.cer>` on Linux is the path and name of the Certificate Authority's root certificate for the LDAP server. The file extension may be `.cer` rather than `.crt`. You can also use a different value for `-alias`.

2. At the prompt to import the certificate, type **Yes**.
3. Press **Enter**.
4. Restart Codar service, see "Restart Codar" on page 60.

Configure secure connections for Oracle database

If the Oracle database server requires a secure connection, complete the following steps (if the Oracle database does not require a secure connection, you can omit these steps):

1. Complete one of the following tasks:
   - If you do not want to configure Codar to check the database DN, complete the following steps:
     i. Open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor.
     ii. Add the following to the Oracle datasource:

     ```
     <connection-url>jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=
     (PROTOCOL = TCPS)(HOST = <host>)(PORT = 1521)))(CONNECT_DATA =(SERVICE_NAME =
     ORCL)))</connection-url>
     ```

     `<host>` is the name of the system on which the Oracle database server is installed.
     iii. Save and close the file.
     iv. Import the Oracle database server Certificate Authority's root certificate into the Java truststore of Codar.
        A. Copy the Oracle database server Certificate Authority's root certificate to the Codar system. If necessary, contact your database administrator to obtain the Oracle database server certificate.
        B. On the Codar system, open a command prompt and run the `keytool` utility with the following options to create a local trusted certificate entry for the Oracle database server.

        On Windows:

        ```
        "CSA_JRE_HOME\bin\keytool" -importcert -trustcacerts
        -alias oracledb
        -keystore "CSA_JRE_HOME\lib\security\cacerts"
        -file <c:\certfile_name.cer> -storepass <password>
        ```

        On Linux:

        ```
         CSA_JRE_HOME
        bin/keytool -importcert -trustcacerts
        -alias oracledb
        -keystore CSA_JRE_HOME/lib/security/cacerts
        -file </tmp/certfile_name.cer> -storepass <password>
        ```

        `CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed.

`<c:\certfile_name.cer>` on Windows or `</tmp/certfile_name.cer>` on Linux is the path and name of the Certificate Authority's root certificate for the LDAP server. The file extension may be `.cer` rather than `.cer`. You can also use a different value for `-alias`.

C. At the prompt to import the certificate, type **Yes**.

D. Press **Enter**.

E. Restart Codar, see "Restart Codar" on page 60.

- If you want to configure Codar to check the database DN, complete the following steps:

i. Open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor.

ii. Add the following to the Oracle datasource:

```
<connection-url>jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=
(PROTOCOL = TCPS)(HOST = <host>)(PORT = 1521)))(CONNECT_DATA =(SERVICE_NAME =
ORCL))(SECURITY=(SSL_SERVER_CERT_
DN="CN=abc,OU=dbserver,O=xyz,L=Sunnyvale,ST=CA,C="US")))</connection-url>
```

`<host>` is the name of the system on which the Oracle database server is installed.

iii. Add the following to the system-properties element:

```
<property name="oracle.net.ssl_server_dn_match" value="true" />
```

iv. Save and close the file.

v. Import the Oracle database server Certificate Authority's root certificate into the Java truststore of Codar.

A. Copy the Oracle database server Certificate Authority's root certificate to the Codar system. If necessary, contact your database administrator to obtain the Oracle database server certificate.

B. On the Codar system, open a command prompt and run the `keytool` utility with the following options to create a local trusted certificate entry for the Oracle database server.

On Windows:

```
"CSA_JRE_HOME\bin\keytool" -importcert -trustcacerts
-alias oracledb
-keystore "CSA_JRE_HOME\lib\security\cacerts"
-file <c:\certfile_name.cer> -storepass <password>
```

On Linux:

```
 CSA_JRE_HOME
bin/keytool -importcert -trustcacerts
-alias oracledb
-keystore CSA_JRE_HOME/lib/security/cacerts
-file </tmp/certfile_name.cer> -storepass <password>
```

`CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed.

`<c:\certfile_name.cer>` on Windows or `</tmp/certfile_name.cer>` on Linux is the path and name of the Certificate Authority's root certificate for the LDAP server. The file extension may be `.cer` rather than `.crt`. You can also use a different value for `-alias`.

C. At the prompt to import the certificate, type **Yes**.

D. Press **Enter**.

E. Restart Codar, see "Restart Codar" on page 60.

2. If client authentication is enabled on the Oracle database server, complete the following steps:

   a. Open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor.

   b. Add the following to the `system-properties` element:

   ```
   <property name="javax.net.ssl.keyStore" value="<certificate_key_file>" />
   <property name="javax.net.ssl.keyStorePassword" value="<certificate_key_file_
   password>" />
   <property name="javax.net.ssl.keyStoreType" value="<certificate_key_file_type>"
   />
   ```

   `<certificate_key_file>` is the same keystore file defined by the `certificate-key-file` attribute in the `ssl` element (for example, `CSA_HOME\jboss-as\standalone\configuration\.keystore` on Windows or `CSA_HOME/jboss-as/standalone/configuration/.keystore` on Linux).

   `<certificate_key_file_password>` is the password to the keystore file.

   `<certificate_key_file_type>` is the keystore type (for example, JKS or PKCS12).

   c. Save and close the file.

   d. Use Oracle's wallet manager to import Codar's certificate into the Oracle database server's wallet as a trusted certificate.

Configure secure connections for Microsoft SQL server

If Microsoft SQL Server requires a secure connection, complete the following steps (if Microsoft SQL Server does not require a secure connection, you can omit these steps):

1. Open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor.
2. Locate the `connection-url` entry for the Microsoft SQL Server datasource and change `ssl=request` to `ssl=authenticate`.

   For example:

   ```
   <connection-url>
      jdbc:jtds:sqlserver://127.0.0.1:1433/example;ssl=~~request~~authenticate
   </connection-url>
   ```
3. Save and close the file.
4. Import the Microsoft SQL Server Certificate Authority's root certificate into the Java truststore of Codar.

   a. Copy the Microsoft SQL Server Certificate Authority's root certificate to the Codar system. If necessary, contact your database administrator to obtain the Microsoft SQL Server certificate.

   b. On the Codar system, open a command prompt and run the `keytool` utility with the following options to create a local trusted certificate entry for the Microsoft SQL Server.

   On Windows:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -trustcacerts
   -alias mssqldb -keystore "CSA_JRE_HOME\lib\security\cacerts"
   -file <c:\certfile_name.cer> -storepass <password>
   ```

On Linux:

```
CSA_JRE_HOME/bin/keytool -importcert -trustcacerts
-alias mssqldb -keystore CSA_JRE_HOME/lib/security/cacerts
-file </tmp/certfile_name.cer> -storepass <password>
```

`CSA_JRE_HOME` is the directory in which the JRE that is used by Codar is installed.

`<c:\certfile_name.cer>` on Windows or `</tmp/certfile_name.cer>` on Linux is the path and name of the Certificate Authority's root certificate for the LDAP server. The file extension may be `.cer` rather than `.crt`. You can also use a different value for `-alias`.

c. At the prompt to import the certificate, type **Yes**.

d. Press **Enter**.

e. Restart Codar, see "Restart Codar" on page 60.

Configure secure connections for Operations Orchestration Load Balancer

If the Operations Orchestration Load Balancer server requires a secure connection, follow these steps to import the Operations Orchestration Load Balancer server Certificate Authority's root certificate into the Java truststore of Codar. If necessary, contact your Operations Orchestration Load Balancer administrator to obtain the Operations Orchestration Load Balancer server certificate.

For each system running Codar, import the root certificate of Operations Orchestration Load Balancer's Certificate Authority into Codar (you must first export Operations Orchestration Load Balancer's certificate from Operations Orchestration Load Balancer's truststore and then import it into Codar's truststore).

1. Open Operations Orchestration Load Balancer in a Web browser (using https).
2. Export the certificate from the Web browser.

   If you are using a Chrome web browser, complete the following steps:

   a. In the address bar, click the lock icon with the red X over it and select **certificate information**.

   b. In the Certificate dialog, do the following:

      i. Select the **Details** tab.

      ii. Click **Copy to File**.

      iii. In the Certificate Export Wizard, do the following:

         A. Click **Next**.

         B. Select **Base-64 encoded X.509 (.CER)** and click **Next**.

         C. Click **Browse** and select a directory in which to save the certificate.

            - If you are running Operations Orchestration Load Balancer on the same system as Codar, select the `CSA_JRE_HOME\lib\security` directory, enter **paslb.cer** as the file name, and click **Save**.
            - If you are running Operations Orchestration Load Balancer on a system that is not running Codar, select a directory in which to store the certificate file, enter **paslb.cer** as the file name, and click **Save**.

         D. Click **Next**.

         E. Click **Finish**.

         F. Click **OK**.

   iv. Click **OK**.

If you are using a Firefox web browser, complete the following steps:

a. Click **Add Exception**.
b. In the Add Security Exception dialog, click **View**.
c. In the Certificate Viewer, do the following:
   i. Select the **Details** tab.
   ii. Click **Export**.
   iii. Select a directory in which to save the certificate.
      - If you are running Operations Orchestration Load Balancer on the same system as Codar, select the `CSA_JRE_HOME\lib\security` directory, enter **paslb.cer** as the file name, select **X.509 Certificate (PEM)** as the Type, and click **Save**.
      - If you are running Operations Orchestration Load Balancer on a system that is not running Codar, select a directory in which to store the certificate file, enter **paslb.cer** as the file name, select **X.509 Certificate (PEM)** as the Type, and click **Save**.
   iv. Click **Close**.
   v. Click **Cancel**.

If you are using a Windows IE web browser, complete the following steps:

a. In the address bar, click **Certificate Error** and select **View certificates**.
b. In the Certificate Export Wizard, do the following:
   i. Select the **Details** tab.
   ii. Click **Copy to File**.
   iii. In the Certificate Export Wizard, do the following:
      A. Click **Next**.
      B. Select **Base-64 encoded X.509 (.CER)** and click **Next**.
      C. Click **Browse** and select a directory in which to save the certificate.
         - If you are running Operations Orchestration Load Balancer on the same system as Codar, select the `CSA_JRE_HOME\lib\security` directory, enter **paslb.cer** as the file name, and click **Save**.
         - If you are running Operations Orchestration Load Balancer on a system that is not running Codar, select a directory in which to store the certificate file, enter **paslb.cer** as the file name, and click **Save**.
      D. Click **Next**.
      E. Click **Finish**.
      F. Click **OK**.
   iv. Click **OK**.

3. If you are running Operations Orchestration Load Balancer on a system that is not running Codar, copy the `paslb.cer` file to the `CSA_JRE_HOME\lib\security` directory on the system running Codar.
4. On the system running Codar, open a command prompt and run the following commands:

   Windows:

   ```
   cd "CSA_JRE_HOME\lib\security"
   ```

```
..\..\bin\keytool -importcert -alias paslb -file paslb.cer
-keystore cacerts -storepass <password>
```

Linux:

```
cd CSA_JRE_HOME/lib/security

../../bin/keytool  -importcert -alias paslb -file paslb.cer
-keystore cacerts -storepass <password>
```

5. When prompted to trust the certificate, enter **yes**.

Operations Orchestration

The Codar solution includes a number of Operations Orchestration flows that perform Codar operations.

> **Note:** If you followed the instructions in the *Codar Installation and Configuration Guide* or *Codar Upgrade Guide* to configure Operations Orchestration, you should have already completed the tasks in this section to configure Operations Orchestration.

In this release, you can install Operations Orchestration with Codar using the Codar installer or you can install Operations Orchestration externally. Only one instance of Operations Orchestration is required for both topology and sequential designs. If you have upgraded from an earlier version of Codar, you may have configured multiple instances of Operations Orchestration for sequential designs. If you have upgraded from an earlier version of Codar that uses multiple instances of Operations Orchestration for sequential designs, you can continue to use the multiple instances of Operations Orchestration for sequential designs. If you have upgraded from an earlier version of Codar that uses only a single instance of Operations Orchestration or are installing Codar for the first time, only one configured instance of Operations Orchestration is supported.

> Codar includes by default a 90 day trial license of Operations Orchestration. After 90 days, you must install the Operations Orchestration license.

This chapter describes the following tasks:

- Install the Operations Orchestration license
- "Configure Operations Orchestration for topology designs" on the next page
- "Integrate with Operations Orchestration " on page 47

Apply the Operations Orchestration license

After 90 days, the Operations Orchestration license that is packaged with Codar expires and you must apply a new license.

You must contact HPE Customer Support to acquire the new license. After HPE Customer Support provides the new license, download it on your system.

To apply the Operations Orchestration license:

1. Log on to Operations Orchestration.
2. Click **System Configuration** on the left pane.
3. Click the **System Settings** tab.
4. On the **License** tab, click the **Install License** button.
5. You are prompted to select the license file. Browse to the path in which you downloaded the license file and select it.
6. Click **OK**.

The Operations Orchestration license is installed.

Configure Operations Orchestration for topology designs

The following tasks are to configure Operations Orchestration for topology designs. Configure only one instance of Operations Orchestration for topology designs.

> **Note:** If you followed the instructions in the *Codar Installation and Configuration Guide* or *Codar Upgrade Guide* to configure Operations Orchestration, you should have already completed the tasks in this section to configure Operations Orchestration.

Complete the following tasks to configure Operations Orchestration to integrate with Codar:

- "Configure internal user" below
- "Deploy content packs" on the next page
- "Configure Single Sign-On between Codar and Operations Orchestration" on the next page
- "Configure Operations Orchestration properties in csa.properties file" on page 43
- "Configure secure connection between Codar and Operations Orchestration" on page 44
- "Run component tool" on page 44

> **Note:** In the following instructions, `CSA_HOME` is the directory in which Codar is installed and `ICONCLUDE_HOME` is where you installed Operations Orchestration.
>
> Be sure all the latest patches for Operations Orchestration have been installed. See the *Codar System and Software Support Matrix*.

Configure internal user

Internal users can be used to configure Operations Orchestration for Codar. The user in these instructions is used for provisioning topology designs.

To configure an internal user, complete the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **System Configuration** button.
3. Select **Security** > **Internal Users**.
4. Click the **Add** button.
5. Enter the following information:

Field	Recommended value
User Name	admin
Password	cloud
Roles	ADMINISTRATOR, SYSTEM, ADMIN

   The admin user is used with Single Sign-On. When Operations Orchestration is launched from the Codar Console, this user allows access to Operations Orchestration without having to log in. If you are using topology designs, the admin user can also be used for provisioning topology designs.

6. Click **Save**.

7. Enable authentication by selecting the Enable Authentication check box.
8. Select **OK** in the confirmation dialog.

Deploy content packs

1. From Operations Orchestration Central, click the **Content Management** button.
2. Click the **Content Packs** tab.
3. Click the **Deploy New Content** icon.
4. In the Deploy New Content dialog, click the **Add files for deployment** icon.
5. Click the **Deploy New Content** icon.
6. Click the Add files for deployment icon.
7. Navigate to the `CSA_HOME\Tools\ComponentTool\contentpacks\` directory, select all the content packs, and click **Open**.
8. Click **Deploy**.

   The deployment may take a few minutes and the dialog will show a progress bar.
9. When the deployment succeeds, click Close to close the dialog.

Configure Single Sign-On between Codar and Operations Orchestration

If Single Sign-On was enabled during installation of Codar, Single Sign-On can be configured between Codar and Operations Orchestration. Configuring Single Sign-On allows you to launch Operations Orchestration from the Codar Console without having to log in to Operations Orchestration.

Codar provides an out-of-the-box user (admin) and password (cloud) and, earlier in this guide, you configured an internal user for Operations Orchestration with the same username and password. When Single Sign-On is configured between Codar and Operations Orchestration, this user can be used for single sign-on. That is, if you are logged in to Codar as the admin user, you can launch Operations Orchestration from the Codar Console and not have to log in to Operations Orchestration.

You can also configure LDAP users for single sign-on. In order to enable single sign-on for LDAP users, you must either configure Codar and the embedded Operations Orchestration to use the same LDAP source or, if Codar and the embedded Operations Orchestration use different LDAP sources, configure the same users in both sources. In either case, the Codar user must be assigned to the Codar Administrator or Service Operations Manager role and the embedded Operations Orchestration user must be assigned any role that allows flows to be viewed.

> **Note:** In order to use Single Sign-On between Codar and Operations Orchestration, the systems on which Codar and Operations Orchestration are installed must be in the same domain.

Configure and enable Single Sign-On

To configure and enable Single Sign-On on Operations Orchestration, complete the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **System Configuration** button.
3. Select **Security > SSO**.

4. Select the **Enable** check box.
5. Enter the **InitString**. This is the value to which the crypto InitString attribute is set in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEBINF\hpssoConfiguration.xml` file.

   For example, if the entry in the file is `crypto InitString="lOJisF9Slbf79hmLsd"`, copy `lOJisF9Slbf79hmLsd` to this field. This string is used to encrypt and decrypt the `LWSSO_COOKIE_KEY` cookie that is used to authenticate the user for single sign-on.
6. Enter the **Domain**. This is the domain name of the network of the servers on which Codar and Operations Orchestration are installed.
7. Click **Save**.

Configure LDAP users for single sign-on

In order to enable single sign-on for LDAP users, you must either configure Codar and Operations Orchestration to use the same LDAP source or, if Codar and Operations Orchestration use different LDAP sources, configure the same users in both sources. In either case, the Codar user and the Operations Orchestration user must be assigned any role that allows flows to be viewed.

For more information on configuring LDAP in Operations Orchestration, see the *Operations Orchestration Central Help*.

Note: One of the LDAP servers must be set to default in Operations Orchestration so that Codar can launch the Operations Orchestration page. Otherwise, an "access denied" error occurs.

To configure LDAP for Operations Orchestration, complete the following steps:

1. Log in to Operations Orchestration Central.
2. Click the System Configuration button.
3. Select Security > LDAP.
4. Enter the information to configure LDAP.
5. Click **Save**.

Configure Operations Orchestration properties in csa.properties file

If you integrated with Operations Orchestration using the installer (during the installation or upgrade process), you do not need to configure these properties (they are already configured). These properties are used to integrate with Operations Orchestration.

In the subscription event overview section of the **Operations** area in the Codar Console, selecting the Process ID opens Operations Orchestration to the detailed page of the selected process when these properties are configured.

Edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file and configure the following properties:

Property	Description
OOS_URL	The URL used to access Operations Orchestration Central. This is the Operations Orchestration used for provisioning topology designs (Operations Orchestration version 10.21). Set this URL to the system on which Operations Orchestration version 10.21 is

Property	Description
	installed. For example, `https://<hostname>:8443`.
OOS_USERNAME	The username used to log in to Operations Orchestration Central. Set this username to admin.
OOS_PASSWORD	The encrypted password used by the user defined in `OOS_USERNAME` to log in to Operations Orchestration Central. Set this property to the encrypted value of the user defined in `OOS_USERNAME` (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses.
embedded.oo.root.dir	Location of the embedded Operations Orchestration when it is installed with CSA. This property is generated when embedded Operations Orchestration is installed during the CSA installation. This property is the only indicator of embedded Operations Orchestration, which is important mainly for uninstallation and upgrades. This property cannot be edited.

Configure secure connection between Codar and Operations Orchestration

If you integrated with Operations Orchestration using the installer (during the installation or upgrade process), you do not need to configure a secure connection (it has already been configured).

Run component tool

The component tool imports the Operations Orchestration flows from the content packs installed with Codar (used only with Operations Orchestration version 10.21).

To run the component tool, complete the following steps:

1. Open a command prompt and change the directory to `CSA_HOME\Tools\ComponentTool`.
2. Generate the sample database properties files. Run the following command:

   Windows:

   ```
   "CSA_JRE_HOME\bin\java" -jar component-tool.jar -g
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/java -jar component-tool.jar -g
   ```

3. Make a copy of the appropriate sample database properties file, rename it to `config.properties`, and update the content, as needed.

Property Name	Description
jdbc. driver ClassNam	The JDBC driver class. **Example**

Property Name	Description

e	`Oracle: jdbc.driverClassName=oracle.jdbc.driver.OracleDriver`
	`MS SQL: jdbc.driverClassName=net.sourceforge.jtds.jdbc.Driver`
	`PostgreSQL: jdbc.driverClassName=org.postgresql.Driver`

jdbc.dialect	The classname that allows JDBC to generate optimized SQL for a particular database.
	Example
	`Oracle: jdbc.dialect=org.hibernate.dialect.OracleDialect`
	`MS SQL: jdbc.dialect=org.hibernate.dialect.SQLServerDialect`
	`PostgreSQL: jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect`

jdbc.databaseUrl	The JDBC URL. When specifying an IPv6 address, it must be enclosed in square brackets (see example below).
	Example
	Oracle, TLS not enabled
	`jdbc.databaseUrl=jdbc:oracle:thin:@127.0.0.1:1521:XE`
	
	Oracle, TLS not enabled, using an IPv6 address
	`jdbc.databaseUrl=jdbc:oracle:thin:@[f000:253c::9c10:b4b4]:1521:XE`
	
	Oracle, TLS enabled, Codar does not check the database DN
	`jdbc.databaseUrl=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL = TCPS)(HOST = <host>)(PORT = 1521))) (CONNECT_DATA =(SERVICE_NAME = ORCL)))`
	where `<host>` is the name of the system on which the Oracle database server is installed.
	
	Oracle, TLS enabled, Codar checks the database DN
	`jdbc.databaseUrl=jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCPS)(HOST = <host>)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME = ORCL))(SECURITY=(SSL_SERVER_CERT_DN="CN=abc,OU=dbserver,O=xyz,L=Sunnyvale,ST=CA,C=US")))`
	where `<host>` is the name of the system on which the Oracle database server is installed and the values for `SSL_SERVER_CERT_DN` are for the DN of the Oracle database server.
	
	MS SQL, TLS not enabled
	`jdbc.databaseUrl=jdbc:jtds:sqlserver://127.0.0.1:1433/example;ssl=request`
	
	MS SQL, TLS not enabled, using an IPv6 address
	`jdbc.databaseUrl=jdbc:jtds:sqlserver://[::1]:1433/example;ssl=request`
	
	MS SQL, TLS enabled
	`jdbc.databaseUrl=jdbc:jtds:sqlserver://127.0.0.1:1433/example;ssl=authenticate`

Property Name	Description
	PostgreSQL `jdbc.databaseUrl=jdbc:postgresql://127.0.0.1:5432/codardb`
jdbc. username	The user name of the database user you configured for Codar after installing the database.
jdbc. password	The password for the database user. The password should be encrypted (see the "Encrypt password" on page 60 for instructions on encrypting passwords). **Example** `jdbc.password=ENC(fc5e38d38a5703285441e7fe7010b0)`

Example `config.properties` content

Oracle, TLS not enabled

```
jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
jdbc.dialect=org.hibernate.dialect.OracleDialect
jdbc.databaseUrl=jdbc:oracle:thin:@127.0.0.1:1521:XE
jdbc.username=codar
jdbc.password=ENC(fc5e38d38a5703285441e7fe7010b0)
```

MS SQL, TLS not enabled

```
jdbc.driverClassName=net.sourceforge.jtds.jdbc.Driver
jdbc.dialect=org.hibernate.dialect.SQLServerDialect
jdbc.databaseUrl=jdbc:jtds:sqlserver://127.0.0.1:1433/example;ssl=request
jdbc.username=codar
jdbc.password=ENC(fc5e38d38a5703285441e7fe7010b0)
```

MS SQL, TLS enabled

```
jdbc.driverClassName=net.sourceforge.jtds.jdbc.Driver
jdbc.dialect=org.hibernate.dialect.SQLServerDialect
jdbc.databaseUrl=jdbc:jtds:sqlserver://127.0.0.1:1433/example;ssl=authenticate
jdbc.username=codar
jdbc.password=ENC(fc5e38d38a5703285441e7fe7010b0)
```

PostgreSQL

```
jdbc.driverClassName=org.postgresql.Driver
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
jdbc.databaseUrl=jdbc:postgresql://127.0.0.1:5432/codardb
jdbc.username=codardbuser
jdbc.password=ENC(fc5e38d38a5703285441e7fe7010b0)
```

4. Run the component tool:

   - **Oracle**

     Windows:

     ```
     "CSA_JRE_HOME\bin\java" -jar component-tool.jar -c config.properties
     ```

```
-cp contentpacks -m mappingFiles -me metainfo.txt -j <jdbc_driver_
directory>\ojdbc.jar
```

Linux:

```
CSA_JRE_HOME/bin/java -jar component-tool.jar -c config.properties
-cp contentpacks -m mappingFiles -me metainfo.txt -j <jdbc_driver_
directory>/ojdbc.jar
```

- **MS SQL and PostgreSQL**

  Windows:

  ```
  "CSA_JRE_HOME\bin\java" -jar component-tool.jar -c config.properties
  -cp contentpacks -m mappingFiles -me metainfo.txt
  ```

  Linux:

  ```
  CSA_JRE_HOME/bin/java -jar component-tool.jar -c config.properties
  -cp contentpacks -m mappingFiles -me metainfo.txt
  ```

> **Note:** Do not edit the `metainfo.txt` file or the `contentpacks` and `mappingFiles` directories.

Integrate with Operations Orchestration

Complete the following tasks to configure Operations Orchestration to integrate with Codar:

- "Add JRE to system path" below
- "Install Codar content pack " on the next page
- "Configure internal users" on the next page
- "Deploy content packs required by Codar" on page 49
- "Set up system accounts for Codar content pack" on page 50
- "Set up system properties forCodar content pack" on page 50
- "Configure Single Sign-On between Codar and Operations Orchestration" on page 51
- "Configure secure connection between Codar and Operations Orchestration" on page 52

> **Note:** In the following instructions, `CSA_HOME` is the directory in which Codar is installed and `ICONCLUDE_HOME` is where you installed Operations Orchestration.
>
> Be sure all the latest patches for Operations Orchestration have been installed. See the *Codar System and Software Support Matrix*.

Add JRE to system path

The flows that are imported require that a JRE be included in the system path on the system running Codar.

To add a JRE to the system path on Windows, complete the following steps:

1. Open the **Environment Variables** dialog:
   a. Right-click **Computer** and select **Properties**.
   b. Select **Advanced System Settings**.
   c. Click **Environment Variables**.

2. Select the **Path** system variable.
3. Click **Edit**.
4. At the end of the value for **Variable value**, add a semicolon (;) and the following path:

   If Operations Orchestration and Codar are installed on the same system:

   ```
   ICONCLUDE_HOME\java\bin
   ```

   or

   If Operations Orchestration and Codar are installed on different systems:

   ```
   CSA_JRE_HOME\bin
   ```
5. Click **OK** and close all windows.

To add a JRE to the system path on Linux, complete the following steps:

Open a shell and enter one of the following commands:

- If Operations Orchestration and Codar are installed on the same system, enter this command:

  ```
  export PATH=$PATH:$ICONCLUDE_HOME/java/bin
  ```
- If Operations Orchestration and Codar are installed on different systems, enter this command:

  ```
  export PATH=$PATH:$CSA_JRE_HOME/bin
  ```

> **Note:** By setting the system path, all applications (that require a JRE) use the JRE that is installed with Operations Orchestration or Codar (depending on the path you configured and if it is the only path or the first path set to a JRE in the system path). If you need to run another JRE with an application, you must type in the relative path to that JRE in order to run it (for example, when you configure TLS).

Install Codar content pack

If Codar and Operations Orchestration are running on different systems, copy the `CSA_HOME\CSAKit-4.5\OO Flow Content\10X\oo10-csa-cp-4.50.0000.jar` file from the Codar system to the Operations Orchestration system (where `CSA_HOME` is the directory in which Codar is installed).

Configure internal users

Internal users can be used to configure Operations Orchestration for Codar.

To configure an internal user, complete the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **System Configuration** button.
3. Select **Security** > **Internal Users**.
4. Click the Add (**+**) icon.
5. Enter the following information:

Field	Recommended value
User Name	codaroouser
Password	cloud

Field	Recommended value
Roles	ADMINISTRATOR, SYSTEM, ADMIN

The codaroouser user is used to import the Operations Orchestration flows. When importing flows, this user is configured in the Operations Orchestration input file used by the process definition tool.

6. Click **Save**.
7. Enable authentication by selecting the **Enable Authentication** check box.
8. Click **OK** in the confirmation dialog.
9. Click the **Add** button.
10. Enter the following information:

Field	Recommended value
User Name	admin
Password	cloud
Roles	ADMINISTRATOR, SYSTEM, ADMIN

The admin user is used with Single Sign-On. When Operations Orchestration is launched from the Codar Console, this user allows access to Operations Orchestration without having to log in. If you are using topology designs, the admin user can also be used for provisioning topology designs.

11. Click **Save**.
12. Enable authentication by selecting the **Enable Authentication** check box.
13. Click **OK** in the confirmation dialog.
14. Log out of Operations Orchestration Central and log back in as the codaroouser.

Deploy content packs required by Codar

To deploy content packs required by Codar, complete the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **Content Management** button.
3. Click the **Content Packs** tab.
4. Click the **Deploy New Content** icon.
5. In the **Deploy New Content** dialog, click the **Add files for deployment** icon.
6. Click the **Deploy New Content** icon.
7. Click the **Add files for deployment** icon.
8. Navigate to the `CSA_HOME/CSAKit-4.5/OOFlowContent/10X` directory, select all content packs to be deployed, and click **Open**.
9. Click **Deploy**.

   The deployment may take a few minutes and the dialog will show a progress bar.
10. When the deployment succeeds, click **Close** to close the dialog.

Set up system accounts for Codar content pack

Set up system accounts for the Codar content pack by completing the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **Content Management** button.
3. Select **Configuration Items > System Accounts**.
4. Click the Add (**+**) icon.
5. Enter the following information if it is not already configured:

Field	Recommended value
System Account Name	`CSA_REST_CREDENTIALS`
User Name	ooInboundUser
Passwords	cloud

> **Note:** The **User Name** configured for the `CSA_REST_CREDENTIALS` System Account setting must match the **Override Value** (Operations Orchestration version 10.21) configured for the `CODAR_OO_USER` System Property setting.

6. Click **Save**.
7. Click the **Add** icon.
8. Enter the following information if it is not already configured:

Field	Recommended value
System Account Name	`CSA_SERVICEMANAGER_CREDENTIALS`
User Name	falcon
Passwords	*<leave_blank>*_

9. Click **Save**.

Set up system properties forCodar content pack

Set up the following system properties for the Codar content pack by completing the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **Content Management** button.
3. Select **Configuration Items > System Properties**.
4. Click **the** Add icon.

5. Enter the following information if it is not already configured:

Field	Recommended value
Name	`CSA_REST_URI`
Override Value	https://<*codar_hostname*>:8444/csa/rest

6. Click **Save**.

Configure Single Sign-On between Codar and Operations Orchestration

If Single Sign-On was enabled during installation of Codar, Single Sign-On can be configured between Codar and Operations Orchestration. Configuring Single Sign-On allows you to launch Operations Orchestration from the Codar Console without having to log in to Operations Orchestration.

Codar provides an out-of-the-box user (admin) and password (cloud) and, earlier in this guide, you configured an internal user for Operations Orchestration with the same username and password. When Single Sign-On is configured between Codar and Operations Orchestration, this user can be used for single sign-on. That is, if you are logged in to Codar as the admin user, you can launch Operations Orchestration from the Cloud Service Management Console and not have to log in to Operations Orchestration.

You can also configure LDAP users for single sign-on. In order to enable single sign-on for LDAP users, you must either configure Codar and the embedded Operations Orchestration to use the name LDAP source or, if Codar and the embedded Operations Orchestration use different LDAP sources, configure the same users in both sources. In either case, the Codar user must be signed to the Codar Administrator or Service Operations Manager role and the embedded Operations Orchestration user must be assigned any role that allows flows to be viewed.

> **Note:** In order to use Single Sign-On between Codar and Operations Orchestration, the systems on which Codar and Operations Orchestration are installed must be in the same domain.

Configure and enable Single Sign-On

To configure and enable Single Sign-On on Operations Orchestration, complete the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **System Configuration** button.
3. Select **Security > SSO**.
4. Select the **Enable** check box.
5. Enter the **InitString**. This is the value to which the `crypto InitString` attribute is set in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\hpssoConfiguration.xml` file. For example, if the entry in the file is `cryptoInitString="lOJisF9Slbf79hmLsd"`, copy `lOJisF9Slbf79hmLsd` to this field. This string is used to encrypt and decrypt the `LWSSO_COOKIE_KEY` cookie that is used to authenticate the user for single sign-on.
6. Enter the **Domain**. This is the domain name of the network of the servers on which Codar and Operations Orchestration are installed.
7. Click **Save**.

Configure LDAP users for single sign-on

In order to enable single sign-on for LDAP users, you must either configure Codar and Operations Orchestration to use the same LDAP source or, if Codar and Operations Orchestration use different LDAP sources, configure the same users in both sources. In either case, the Codar user and the Operations Orchestration user must be assigned any role that allows flows to be viewed.

For more information on configuring LDAP in Operations Orchestration, see the *Operations Orchestration Central Help*.

Note: One of the LDAP servers must be set to default in Operations Orchestration so that Codar can launch the Operations Orchestration page. Otherwise, an "access denied" error occurs.

To configure LDAP for Operations Orchestration complete the following steps:

1. Log in to Operations Orchestration Central.
2. Click the **System Configuration** button.
3. Select **Security > LDAP**.
4. Enter the information to configure LDAP.
5. Click **Save**.

Configure secure connection between Codar and Operations Orchestration

If you integrated with Operations Orchestration using the installer (during the installation or upgrade process), you do not need to configure a secure connection (it has already been configured).

Codar Console

This chapter provides information about the tasks needed to prepare and set up the Codar Console in order to start using Codar. You must complete the required tasks before you can start to use the Codar Console. Organization roles provide authorization for members to perform these tasks.

The roles and tasks are included in the following topics:

- "Roles in Codar" below
- "Configure provider organization" on the next page (required)
- "Add software license" on page 55 (required)
- "Proxy configuration for resource providers outside the internal network" on page 55 (optional)
- "Customize dashboard" on page 1 (optional)
- "Customize Codar Console title" on page 57 (optional)

Roles in Codar

Codar users with the following roles can create roles and assign permissions that they want to each role:

- Administrator
- Application Architect
- Application Release Manager

For information about creating, editing, and deleting roles, see "Roles in Codar" in the Codar Console Help.

Codar also has out-of-the-box roles that are configured and assigned by the administrator. Users with the Administrator role have access to all areas.

Application Architect

Users with this role can

- Create packages.
- View packages in any stage.
- Deploy, update, and delete packages in Development stage only.
- Embrace components.
- Create, update, and delete applications and application versions.

Users with this role cannot promote or reject packages in any stage.

Application Developer

Users with this role can:

- Create packages.
- View packages in any stage.

- Deploy, update, and delete packages in Development stage only.
- Promote packages from Development to Testing stage.

Application QA

Users with this role can:

- View packages in any stage.
- Deploy, update, reject, and delete packages in Testing stage.
- Promote packages from Testing to Staging stage.

Application Release Manager

Users with this role can:

- View packages in any stage.
- View Pipeline Statistics.
- Deploy, update, reject, and delete packages in Staging stage.
- Promote packages from Staging to Production stage.
- Deploy, update, reject, and delete packages in Production stage.

Codar Integration User

Users with this role:

• Can be used to integrate Codar with external systems.

Application Operations manager

Users with this role can:

- View packages in any stage.
- Deploy, reject, edit, and delete packages in Staging stage.
- Promote packages from Staging to Production stage.

Configure provider organization

1. Launch the Codar Console by typing the following URL in a supported web browser: `https://`*`<codarhostname>`*`:8444/csa` where *`<codarhostname>`* is the fully-qualified domain name of the system on which the Codar Console resides.
2. Log in to the Codar Console as an Administrator (see the *Codar Concepts Guide* and *Codar Console Help* for more information about the Codar Administrator role).
3. Click the **Organizations** tile.

   In the left-navigation frame, the provider organization icon appears to the right of the provider organization that is automatically set up (CODAR-Provider). You may modify the provider organization, as needed. However, you cannot delete it. There can be only one provider organization.

4. In the left-navigation frame, select the provider organization.
5. Configure the provider organization by selecting and entering information into each section of the organization's navigation frame (General Information, LDAP, Access Control, Email Notifications, and Catalogs). For details about the fields in each section, see the *Codar Console Help*.

Add software license

Codar version 1.60 requires a software license. Codar licensing is based on the number of operating system instances (OSIs) being used in current, active subscriptions.

After initial installation of Codar version 1.60, when you log in to the Codar Console, a temporary 90-day trial license is activated. Once the trial license expires, you are limited to 25 OSIs. If you created more than 25 OSIs during the trial period, you cannot create any additional OSIs. You can add more licenses at any time to increase your OSI capacity.

After you upgrade to Codar version 1.60, when you log in to the Codar Console, all licenses of earlier versions of Codar are valid and are automatically added.

Before you can add a software license, you must request a license using the licensing portal. See "Request software licenses" on page 13.

For more information about managing Codar licenses, see the *Codar Console Help*.

Proxy configuration for resource providers outside the internal network

If you are using a network proxy server to communicate with a resource provider outside of the internal network (the resource provider's service access point is located outside of the internal network), configure Codar and Operations Orchestration to use this proxy server.

If you are using a network proxy server to communicate with a resource provider outside of the internal network, proxy configuration is required in the following situations:

- Codar - Validating the accessibility of a resource provider's URL. When a resource provider is created or modified, accessibility of the provider URL is validated with an HTTP or HTTPS GET call.
- Operations Orchestration - Contacting a resource provider. When an Operations Orchestration workflow provisioning step is executed, Operations Orchestration attempts to contact the resource provider.

If you do not configure the proxy server, you may see a Provider Validation Failed message when creating or updating a resource provider whose service access point is located outside of the internal network. Or, provisioning of a design fails when Operations Orchestration is unable to communicate with a resource provider that is located outside of the internal network.

To configure the proxy server for Codar and Operations Orchestration, complete the following steps:

1. On the system running Codar, in a text editor, open the `CSA_HOME\jboss-as\bin\standalone.conf.bat` file on Windows or .`CSA_HOME/jboss-as/bin/standalone.conf` file on Linux.
2. After the last uncommented line that sets the JAVA_OPTS property, add the following lines:

   On Windows:

```
rem # HTTP Proxy Settings
set "JAVA_OPTS=%JAVA_OPTS% -Dhttp.proxyHost=<proxy.company.com>
-Dhttp.proxyPort=<proxy_port>"

rem # HTTPS Proxy Settings
set "JAVA_OPTS=%JAVA_OPTS% -Dhttps.proxyHost=<proxy.company.com>
-Dhttps.proxyPort=<proxy_port>"

rem # HTTP/HTTPS hosts not handled by the proxy
set "JAVA_OPTS=%JAVA_OPTS% -
Dhttp.nonProxyHosts=mycodarserver^^^|localhost^^^|127.*^^^|10.* "
```

where <*proxy.company.com*> is the fully-qualified domain name of the proxy server, <*proxy_port*> is the port used to communicate with the proxy server, and `^^^|` is the separator used when defining more than one non-proxy host.

On Ubuntu Linux:

```
 # HTTP Proxy Settings
JAVA_OPTS=$JAVA_OPTS -Dhttp.proxyHost=<proxy.company.com>
-Dhttp.proxyPort=<proxy_port>"

# HTTPS Proxy Settings
JAVA_OPTS=$JAVA_OPTS -Dhttps.proxyHost=<proxy.company.com>
-Dhttps.proxyPort=<proxy_port>"

# HTTP/HTTPS hosts not handled by the proxy
JAVA_OPTS=$JAVA_OPTS -Dhttp.nonProxyHosts=mycodarserver\|localhost\|127.*|10.*"
```

<*proxy.company.com*> is the fully-qualified domain name of the proxy server, <*proxy_port*> is the port used to communicate with the proxy server, and `^^^|` on Windows or `\|` on Linux is the separator used when defining more than one non-proxy host.

Red Hat Enterprise Linux

In the if-else block, add the following lines:

```
# HTTP Proxy Settings
JAVA_OPTS= "$JAVA_OPTS -Dhttp.proxyHost=<proxy.company.com>
-Dhttp.proxyPort=<proxy_port>"

# HTTPS Proxy Settings
JAVA_OPTS= "$JAVA_OPTS -Dhttps.proxyHost=<proxy.company.com>
-Dhttps.proxyPort=<proxy_port>"

# HTTP/HTTPS hosts not handled by the proxy
JAVA_OPTS= "$JAVA_OPTS -Dhttp.nonProxyHosts=localhost\|127.*\|10.* "
```

<*proxy.company.com*> is the fully-qualified domain name of the proxy server, <*proxy_port*> is the port used to communicate with the proxy server, and `\|` is the separator used when defining more than one non-proxy host.

3. Save and exit the file.
4. Restart Codar service, see "Restart Codar" on page 60.
5. If you have integrated with Operations Orchestration version 10.21, do the following:

a. Log in to Operations Orchestration Central.

b. Click the **Content Management** button.

c. Select **Configuration Items** > **System Properties**.

d. Click the **Add** icon.

e. Enter the following information if it is not already configured:

Field	Description
Name	`CODAR_Proxy_Host`
Override Value	The fully-qualified domain name of the proxy server.
Name	`CODAR_Proxy_Port`
Override Value	The port used to communicate with the proxy server.

f. Click **Save**.

Customize Codar Console font

The font used by the Codar Console can be customized. You can change the font if you are a user who has access to the system on which Codar is running. To change the font, on the system running Codar, do the following:

1. Open the `CSA_HOME\jboss-as\standalone\deployments\csa.war\custom\custom.css` file in a text editor.
2. At the end of the file, add the following:

   ```
   html, body {
   font-family: <font_name>;
   }
   ```

   <font_name> is the font used by the Codar Console.

   For example, to change the font to Arial, add the following to the file:

   ```
   html, body {
   font-family: Arial;
   }
   ```
3. Save and exit the file.
4. Restart Codar service, see "Restart Codar" on page 60.

Customize Codar Console title

The Codar Console title appears at the top of the Codar Console next to the HPE logo. By default, the title is "Codar."

You can change the title if you are a user who has access to the system on which Codar is running. To change the title, on the system running Codar, complete the following:

1. Open the `CSA_HOME\jboss-as\standalone\deployments\csa.war\custom\messages.properties` file in a text editor.

2. Add the following attribute and value:

   `codar_title=`*<title>*

   <*title*> is the title that displays at the top of the Codar Console.

   For example, to change the title to CloudSystem," add the following to the file:

   ```
   codar_title=CloudSystem
   ```

   > **Note:** You cannot change the HPE logo.

   If you are translating the title, create a file named `messages_<locale>.properties` instead (where <*locale*> identifies the language to which the title has been translated, for example, `en` for English or `ja` for Japanese).

3. Save and exit the file.

Common Codar tasks

This chapter provides information on how to perform common Codar tasks.

Tasks include:

- "Launch Codar Console" below
- "Start Codar" below
- "Stop Codar"
- "Restart Codar" on the next page
- "Encrypt password" on the next page
- "Clear web browser cache" on page 61
- "Uninstall Codar" on page 61

Launch Codar Console

Launch the Codar Console by typing the following URL in a supported web browser: `https://<codarhostname>:8444/csa` where *<codarhostname>* is the fully-qualified domain name of the system on which the Codar Console resides.

Launch the Codar Console using an IPv6address by typing the following URL in a supported web browser:

```
https://<ipv6_address>:8444/csa/login
```

Start Codar

To start Codar on Windows, complete the following steps:

1. On the server that hosts Codar, navigate to **Start** > **Administrative Tools** > **Services**.
2. Right-click on the Codar service and select **Start**.
3. If you installed an embedded Operations Orchestration instance, right-click on the Operations Orchestration Central service and select **Start**.

To start Codar on Linux, complete the following steps:

1. On the server that hosts Codar, type the following:

   ```
   service codar start
   ```

2. If you installed an embedded Operations Orchestration instance, as the root user (the Operations Orchestration Central service must be started as the root user because an HPEMatrix Operating Environment flow needs to write to the root directory), type:

   ```
   <embeddedOOinstallation>/central/bin/central start
   ```

   For example, type /usr/local/hpe/codar/`OO/central/bin/central start`

Stop Codar

To stop Codar on Windows, complete the following steps:

1. On the server that hosts Codar, navigate to **Start** > **Administrative Tools** > **Services**.
2. Right-click on the Codar service and select **Stop**.
3. If you installed an embedded Operations Orchestration instance, right-click on the Operations Orchestration Central service and select **Stop**.

To stop Codar on Linux, complete the following steps:

1. Type the following command on the server that hosts Codar:

   `service codar stop`
2. If you installed an embedded Operations Orchestration instance, as the root user, type:

   `<embeddedOOinstallation>/central/bin/central stop`.

   For example, type /usr/local/hpe/codar`/OO/central/bin/central stop`

Restart Codar

To restart Codar on Windows, complete the following steps:

1. On the server that hosts Codar, navigate to **Start** > **Administrative Tools** > **Services**.
2. Right-click on the Codar service and select **Restart**.
3. If you installed an embedded Operations Orchestration instance, right-click on the Operations Orchestration Central service and select **Restart**.

To restart Codar on Linux, complete the following steps:

1. On the server that hosts Codar, type the following:

   `service codar restart`
2. If you installed an embedded Operations Orchestration instance, as the root user, type:

   ```
   <embeddedOOinstallation>/central/bin/central stop
   <embeddedOOinstallation>/central/bin/central start.
   ```

   For example, type
   /usr/local/hpe/codar`/OO/central/bin/central stop`
   `/usr/local/hpe/codar/OO/central/bin/central start`

Encrypt password

Encrypt a password for use with Codar configuration only.

To encrypt a password, complete the following steps:

1. Open a command prompt and change to the `CSA_HOME\Tools\PasswordUtil` directory. For example:

   Windows:

   C:\Program Files\HPE\Codar`\Tools\PasswordUtil`

   Linux:

   `/usr/local/hpe/codar/Tools/PasswordUtil`
2. Run the following command:

Windows:

```
"CSA_JRE_HOME\bin\java" -jar passwordUtil-standalone.jar encrypt <myPassword>
```

If you used different names for the keystore, alias, or encrypted symmetric key file, here is an example of the command without using the example names:

```
"CSA_JRE_HOME\bin\java" -jar "CSA_HOME\Tools\PasswordUtil\passwordUtil-
standalone.jar" encrypt <password> JsafeJCE < Codar encryption keystore>
< Codar encryption keystore password>
< Codar encryption keystore alias>
<location and name of the encrypted symmetric key>
```

Note: If you use path separators in the `passwordUtil-standalone.jar` script options, use either a single forward slash (/) or double backward slashes (\\) as your path separator.

Linux:

```
CSA_JRE_HOME/bin/java -jar passwordUtil-standalone.jar encrypt <myPassword>
```

Clear web browser cache

It may be necessary to clear your web browser cache on systems that previously accessed the Codar Console after upgrading Codar.

Uninstall Codar

Uninstalling Codar removes the `CSA_HOME` directory and all of its contents. If all the contents in `CSA_HOME` are not deleted, you must manually delete them and the `CSA_HOME` directory.

If you installed an embedded Operations Orchestration instance with Codar (you installed Operations Orchestration with Codar using the Codar installer), the embedded Operations Orchestration instance is removed. If you are using Codar with an external Operations Orchestration instance (you installed Operations Orchestration separately from Codar), the external Operations Orchestration instance is not removed.

Note: The Codar database is NOT updated or uninstalled.

Uninstall Codar on Windows

To uninstall Codar on Windows, complete the following steps:

1. Log in as the user who installed Codar (for example, `codaruser`).
2. Stop the Codar service:
   a. On the server that hosts Codar, navigate to **Start >Administrative Tools > Services**.
   b. Right-click on the Codar service and select **Stop**.
   c. If you installed an embedded Operations Orchestration instance, right-click on the Operations Orchestration Central service and select **Stop**.
3. Verify that the services were stopped.

If the Codar service is still running, open a command prompt, navigate to `CSA_HOME\jboss-as\bin`, and run the following command:

```
jboss-cli.bat --connect --command=:shutdown
```

4. Close all instances of Windows Explorer, close all command prompts, and exit all programs that are running on the system.
5. Navigate to **Control Panel > Uninstall a program**.
6. Right-click on **HPE Codar** and select **Uninstall/Change**.
7. Click **Uninstall**.
8. Delete the `CSA_HOME` directory and any remaining contents, if they exist.
9. If they exist, delete all Codar entries from the following file:

   ```
   C:\Program Files\Zero G Registry\.com.zerog.registry.xml
   ```

Uninstall Codar on Linux

To uninstall Codar on Linux, complete the following steps:

1. Log in as the user who installed Codar (for example, `codaruser`).
2. Stop the Codar service, by typing:

   ```
   service codar stop
   ```

3. If you installed an embedded Operations Orchestration instance, as the root user, type:

   <embeddedOOinstallation>`/central/bin/central stop`.

   For example, type `/usr/local/hpe/codar/OO/central/bin/central stop`
4. Verify that the services were stopped. For example, if Codar was installed in `/usr/local/hpe/codar`, enter the following:

   ```
   ps -ef | grep /usr/local/hpe/codar
   ps -ef | grep central
   ```

   If there are Codar or Operations Orchestration services running, repeat step 2 or kill the Codar and Operations Orchestration services.
5. Go to the `CSA_HOME/_HPE_CODAR_1_60_0_installation` directory, and enter the following:

   ```
   cd CSA_HOME/_HPE_CODAR_1_60_0_installation
   ```

6. Uninstall Codar. Enter the following:

   ```
   ./Change\ HPE\ Cloud\ Service\ Automation\ Installation
   ```

7. Confirm that you want to uninstall Codar.
8. When uninstallation completes, log in as root and do the following:
   a. If all the contents in `CSA_HOME` are not deleted, you must manually delete them and the `CSA_HOME` directory.
   b. Delete the Codar service scripts. Enter the following:

      ```
      rm /etc/init.d/codar
      ```

   c. If they exist, delete all Codar entries from the following file:

      ```
      /home/codaruser/.com.zerog.registry.xml
      ```

   d. Optionally, remove the `codaruser` user and `codargrp` group.

User administration

This chapter provides information for additional administration and configuration tasks.

Tasks include:

- "Allow non-administrator users to start and stop Codar service on Windows" below (optional)
- "Allow Codar service to be run as non-administrator user on Windows" on page 65 (optional)
- "Change Codar out-of-the-box user accounts for Windows and Linux" on page 67 (optional)

Allow non-administrator users to start and stop Codar service on Windows

When running Codar on Windows, by default, only users with administrator privileges can start or stop the Codar services. This procedure explains how to grant permissions to non-administrator users to start and stop these services. This process involves the following tasks:

- Create a non-administrator user account, if one does not exist.
- Determine the security identifier (SID) of the non-administrator user.
- Set the security descriptor for the services to allow the non-administrator user to start and stop them.
- Add necessary permissions to the Codar installation directory for the non-administrator user.

Allow non-administrator users to start and stop Codar service

To allow non-administrator users to start and stop the Codar service, complete the following steps:

1. Start the Control Panel on the Codar system and click **Add or remove user accounts** that is under **User Accounts**.
2. Click **Create a new account** in the Manage Accounts window that appears.
3. Enter a name for the user, select the **Standard user** radio button if it is not selected, and then click the **Create Account** button to create the user account.

   In this procedure we will use the user account name "CODARUser."
4. Open a command prompt window and run the following command, as is applicable, to display the security descriptor for the Codar service:

   ```
    sc sdshow codar
   ```

   The command returns a security descriptor in Security Descriptor Definition Language (SDDL), like the following example for the Codar service:

   ```
   D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)
   (A;;CCLCSWLOCRRC;;;IU)(A;;;CCLCSWLOCRRC;;;SU)S:
   (AU;FA;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)
   ```
5. Copy the security descriptor that was returned by the above command to a text editor such as Notepad.
6. Run the following command to display the names and SIDs for all existing user accounts:

   ```
   wmic useraccount get name,sid
   ```

7. From the command output, copy the SID for the non-administrator user to the text editor.

   The SID is usually in a format like `S-1-5-21-3637136161-1358011849-3560387905-1014`.

8. Add `(A;;RPWPCR;;;<SID of non-admin user>)` before the `S:(AU;`... portion of the security descriptor that you copied to a text editor earlier in this procedure.

   Using the security descriptor and SID from our example, the result would be as follows, with the added text shown against a gray background:

   ```
   D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)
   (A;;CCLCSWLOCRRC;;;IU)(A;;;CCLCSWLOCRRC;;;SU)(A;;RPWPCR;;;S-1-5-21-3637136161-
   1358011849-3560387905-1014)S:(AU;FA;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)
   ```

9. Run the following command, as is applicable, to set the security descriptor for the Codar service to the new value:

   ```
   -sc sdset codar "<new security descriptor>"
   ```

   This message `[SC] SetServiceObjectSecurity SUCCESS` is returned if the command completes successfully.

Note: Repeat steps 4 through 9 as necessary so that the security descriptor is changed for both services.

Add permissions to the Codar directory for non-administrator user

The non-administrator user now has the permissions necessary to start and stop the Codar service. As a test, you can log in using the non-administrator user account and start and stop the Codar service.

The final steps below will add necessary permissions to the Codar directory for the non-administrator user.

To add permissions to the Codar directory for the non-administrator user, complete the following steps:

1. Log into the Codar machine as administrator.
2. In Windows Explorer, navigate to the Codar installation directory (for example, C:\Program Files\HPE\Codar), right-click on the folder, and select **Properties** in the menu that appears to open the Codar Properties dialog box.
3. Click the **Security** tab in the Codar Properties dialog box.
4. Check if the user is listed in the Group or user names list in the dialog box, and if it is not listed, continue with the next step. If the user is listed, go to Step 7 to continue.
5. Click the **Edit...** button, click the **Add...** button in the dialog box that appears, enter the non-administrator user name in the **Enter the object names to select** field, and then click the **Check Names** button.
6. Select the name, and then click **OK** to add the user to the Group or user names list.
7. Select the user name, select the **Allow** check box for the following permissions, and then click **OK**.
   - Read &execute
   - List folder contents
   - Read
   - Write

Allow Codar service to be run as non-administrator user on Windows

When running Codar on Windows, by default, the Codar service is run as the service user. This section explains how to configure Codar so that the Codar service can be run by non-administrator users. This process involves the following tasks:

- "Create non-administrator users" below
- "Configure Codar service" below
- "Configure file system permissions for non-administrator users" on the next page

Caution: If the Codar service is run as a non-administrator user, you will not be able to do the following:

- Upgrade Codar
- Deploy hotfixes
- Install patches
- Use external tools such as the component tool, content archive tool, database purge tool, process definition tool, provider tool, schema installation tool, and support tool.
- Modify Autopass license data

Note: Certificates must be replaced and regenerated as the Administrator user.

Create non-administrator users

The following example shows how to create two non-administrator user accounts, one for the Codar service to run as and the other for the HPE Marketplace Portal service to run as. Alternatively, but not documented, you may also create a single non-administrator user to run as for both services.

1. Log in as the Administrator.
2. Start the Control Panel on the Codar system and click **Add or remove user accounts** that is under **User Accounts**.
3. Click **Create a new account** in the Manage Accounts window that appears.
4. Enter a name for the user, select the **Standard user** radio button if it is not selected, and then click the **Create Account** button to create the user account.

   Create a user account: **CodarUser**.

Configure Codar service

1. Log in as the Administrator.
2. Stop Codar, see "Stop Codar" on page 59.
3. Back up and then delete the log files in the `CSA_HOME\jboss-as\standalone\log\` directory.
4. Delete all files in the `CSA_HOME\jboss-as\standalone\tmp\` directory.
5. Configure the Codar service to be run as CodarUser:

a. Navigate to **Start** > **Administrative Tools** > **Services**.

b. Right-click on the Codar service and select **Properties**.

c. Select the **Log On** tab.

d. Select **This account**.

e. In the first field, enter **CodarUser**.

f. Enter the password for CodarUser, confirm the password, and click **OK**.

Configure file system permissions for non-administrator users

Assign permissions to each user for the specified directories in the Codar file system.

1. Log in as the Administrator.
2. Open the File Explorer.
3. For each of the directories listed in the following table, do the following (where C:\Program Files\HPE\Codar is the directory in which Codar has been installed):

   a. Right-click on the directory and select **Properties**.

   b. Click the **Security** tab.

   c. Click **Edit**.

   d. Select a user (CodarUser) and select the permissions listed in the table.

   e. Click **OK** to exit the Permissions dialog.

   f. Click **OK** to exit the Properties dialog.

Directory	**User(s)**	**Allowed Permission(s)**
C:\	CodarUser	Full Control Modify Read & execute List folder contents Read Write
C:\Program Files\HPE	CodarUser	Full Control Modify Read & execute List folder contents Read Write
C:\Program Files\HPE\Codar\	CodarUser	Full Control Modify Read & execute List folder contents Read Write
C:\Program Files\HPE\Codar\Autopass	CodarUser	Full Control

Directory	User(s)	Allowed Permission(s)
		Read
C:\Program Files\HPE\Codar\boss-as	CodarUser	Read
C:\Program Files\HPE\Codar\jboss-as\bin	CodarUser	Write
C:\Program Files\HPE\Codar\CONTENT_IMPORT_LOGS	CodarUser	Write
C:\Program Files\HPE\Codar\jboss-as\standalone	CodarUser	Write
C:\Program Files\HPE\Codar\jboss-as\standalone\deployments	CodarUser	Modify Read & execute List folder contents Read Write
C:\Program Files\HPE\Codar\jboss-as\standalone\configuration	CodarUser	Modify Read & execute List folder contents Read Write
C:\Program Files\HPE\Codar\openjre* *This is the JRE used by Codar. If you are using a different JRE, set the permissions to that JRE's directory.	CodarUser	Read & execute List folder contents Read Write
C:\Program Files\HPE\Codar\scripts	CodarUser	Read
C:\Program Files\HPE\Codar\security	CodarUser	Read
C:\Program Files\HPE\Codar\Tools	CodarUser	Read

4. Start Codar, see "Start Codar" on page 59.
5. Examine the `CSA_HOME\jboss-as\standalone\log\server.log` file and verify the changes deployed correctly.

Change Codar out-of-the-box user accounts for Windows and Linux

Codar ships with built-in user accounts. The user accounts are used to authenticate REST API calls and for initial setup and experimentation with the product. For security reasons, you may want to disable or change the passwords associated with these accounts (do not change the user names).

Note: Do not create users in your LDAP directory that match the out-of-the-box users provided by Codar (the out-of-the-box users are `admin`, `csaInboundUser`, `csaCatalogAggregationTransportUser`, `csaReportingUser`, `csaTransportUser`, `idmTransportUser`, `ooInboundUser`, and codarintegrationUser). Creating the same users in LDAP may allow the out-of-the-box users unintended access to the Codar Console or give the LDAP users unintended privileges.

Codar Console user accounts

The following users ship out-of-the-box and are used with the Codar Console:

admin User: Codar Console

Username	admin
Default Password	cloud
Default Role	ROLE_REST
Usage	This account is used to initially log in to the Codar Console to configure the provider organization.
To Disable	Edit the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\classes\csa-provider-users.properties` file. Update the `admin` property to disable this user account. For example, set `admin` to the following value (this value should be encrypted): `cloud,ROLE_REST,disabled` **Note:** This property not only determines if the account is enabled, it also contains the password and the roles that control access to Codar. By default, the unencrypted value of this property is: `cloud,ROLE_REST,enabled` See "Encrypt password" on page 60 for instructions). The encrypted value is preceded by `ENC` without any separating spaces and is enclosed in parentheses. Ensure there is no blank space at the end of the value.
To Change Password	If you change the password to this account, you must update the value of the password in the `csa-provider-users.properties` file and the `securityAdminPassword` property in the `csa.properties` file (you must use the same password). You must also update and use the same password for every REST API call that uses the password. **Updating the admin property in csa-provider-users.properties** Edit the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\classes\csa-provider-users.properties` file. Update the password portion of the `admin` value and encrypt the entire value, including the roles and account status (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses. Ensure there is no blank space at the end of the value. **Note:** This property not only contains the password, but also the roles that control access to Codar and if the account is enabled. By default, the unencrypted value of this property is: `cloud,ROLE_REST,enabled`

admin User: Codar Console, continued

	Updating the securityAdminPassword property in csa.properties Edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file and update the value of the `securityAdminPassword` property. Use the same encrypted password that you entered for the `admin` property in the `csa-provider-users.properties` file. After modifying the `csa.properties` file, restart Codar, see "Restart Codar" on page 60 .

idmTransportUser User: Codar Console

Username	idmTransportUser
Default Password	idmTransportUser
Default Roles	ROLE_AMIN, PERM_IMPERSONATE
Usage	This account is used to authenticate REST API calls.
To Disable	Do not disable this account.
To Change Password	If you change the password to this account, you must update the value of the `securityIdmTransportUserPassword` property in the `csa.properties` file and the `idmTransportUser` property in the `integrationusers.properties` file (you must use the same password) and you must clear the JBoss server and web browser caches. You must also update and use the same password for every REST API call that uses the password. **Updating the securityIdmTransportUserPassword property in csa.properties** Edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file and update the value of the `securityIdmTransportUserPassword` property. Determine a suitable new password (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses. Ensure there is no blank space at the end of the value. **Updating the idmTransportUser property in integrationusers.properties** **Note:** This property not only contains the password, but also the roles that control access to Codar and if the account is enabled. By default, the unencrypted value of this property is: `idmTransportUser,ROLE_ADMIN,PERM_IMPERSONATE,enabled` Edit the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\classes\integrationusers.properties` file and update the value of the `idmTransportUser` property. Use the same password that you used for the `securityIdmTransportUserPassword` property in the `csa.properties` file and encrypt the entire value of the `idmTransportUser` property, including the roles and account status (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses. . Ensure there is no blank space at the

idmTransportUser User: Codar Console, continued

	end of the value.
	Clearing the JBoss server and web browser caches After modifying and saving the changes to the files, clear the JBoss server and web browser caches. To clear the JBoss server cache, remove the contents from the `CSA_HOME\jboss-as\standalone\tmp` directory. See "Clear web browser cache" on page 61 for information on how to clear the web browser cache. **Restarting Codar** After making these changes, restart Codar, see "Restart Codar" on page 60.

ooInboundUser User: Codar Console

Username	ooInboundUser
Default Password	cloud
Default Role	ROLE_REST
Usage	This account is used by Operations Orchestration to authenticate REST API calls with Codar.
To Disable	Do not disable this account.
To Change Password	If you change the password to this account, you must update the value of the password in the `csa-provider-users.properties` file and the `securityOoInboundUserPassword` property in the `csa.properties` file (you must use the same password). You must also update and use the same password for every REST API call that uses the password. **Updating the ooInboundUser property in csa-provider-users.properties** Edit the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\classes\csa-provider-users.properties` file. Update the password portion of the `ooInboundUser` value and encrypt the entire value, including the roles and account status (see "Encrypt password" on page 60 for instructions on how to encrypt this value). The encrypted value is preceded by `ENC` without any separating spaces and is enclosed in parentheses. Ensure there is no blank space at the end of the value. **Note:** This property not only contains the password, but also the roles that control access to Codar and if the account is enabled. By default, the unencrypted value of this property is: `cloud,ROLE_REST,enabled` You must also update and use the same password for the `CSA_REST_CREDENTIALS` system account in Operations Orchestration (located in the Configuration folder of the Public Repository).

ooInboundUser User: Codar Console, continued

	Updating the securityOoInboundUserPassword property in csa.properties If you change the password to this account, you must update the value of the `securityOoInboundUserPassword` property in `csa.properties`. You must also update and use the same password for the `CSA_REST_CREDENTIALS` system account in Operations Orchestration (located in the Configuration folder of the Public Repository). Edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file and update the value of the `securityOoInboundUserPassword` property. Use the same encrypted password that you entered for the `ooInboundUser` property in the `csa-provider-users.properties` file. After modifying the `csa.properties` file, restart Codar, see "Restart Codar" on page 60 .

codarintegrationUser: Codar Console

Username	codarintegrationUser
Default Password	cloud
Default Role	codarintegrationUser
Usage	This account is used in the Jenkins plug-in for integrating with Codar.
To Disable	It is recommended to enable this account so that Jenkins integration will work.
To Change Password	If you change the password to this account, you must update the value of the password in the `csa-provider-users.properties` file and the `securitycodarintegrationUserPassword` property in the `csa.properties` file (you must use the same password). You must also update and use the same password for every REST API call that uses the password. **Updating the codarintegrationUser property in csa-provider-users.properties** Edit the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\classes\csa-provider-users.properties` file. Update the password portion of the `codarintegrationUser` value and encrypt the entire value, including the roles and account status (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses. Ensure there is no blank space at the end of the value. **Note:** This property not only contains the password, but also the roles that control access to Codar and if the account is enabled. By default, the unencrypted value of this property is: `cloud,ROLE_REST,enabled`. You must also update and use the same password for the `CSA_REST_CREDENTIALS` system account in Operations Orchestration (located in the Configuration folder of the Public Repository). **Updating the securitycodarintegrationUserPassword property in csa.properties** If you change the password to this account, you must update the value of the

codarintegrationUser: Codar Console, continued

	`securitycodarintegrationUserPassword` property in `csa.properties`. You must also update and use the same password in `CSA_REST_CREDENTIALS` system account in Operations Orchestration (located in the Configuration folder of the Public Repository). Edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file and update the value of the `securitycodarintegrationUserPassword` property. Use the same encrypted password that you entered for the `codarintegrationUser` property in the `csa-provider-users.properties` file. After modifying the `csa.properties` file, restart Codar, see "Restart Codar" on page 60 .

Note: The codarintegrationUser user account is for the purpose of integrating Codar with external interfaces such as Jenkins. It is highly recommended that you manage this account in LDAP and to do this you need to add this user account to LDAP. For more details, see "Prepare LDAP for Codar" on page 11E.

Configure Account Lockout Mechanism for Codar Console

By default, when the end user attempts to log in to the Codar Console, and enters the wrong password 3 times, the user account is locked out. After 5 minutes, the account is unlocked and the user can attempt to log in again. This section describes the lockout behavior and how to configure the account lockout mechanism.

Lockout Behavior

Following is the account lockout behavior:

- User's account is locked when the wrong password is entered multiple times (configurable).
- When the wrong password is entered, a watch period (configurable) is started during which another wrong password is expected and counted. If during this period the counter reaches the maximum, the account is locked. If the watch period ends before the counter reaches the maximum, the counter is reset.
- Parallel successful authentications during the watch period have no effect on the counter.
- When the account is locked, the user receives *the Invalid User Name* or *Password* message whether the credentials are right or wrong.
- The locked account is unlocked after n minutes (configurable).
- Account locking is not persistent and its state is not synchronized between cluster nodes. Each node is independent, and will forget the locking state upon restart and allow users to log in.

Note: It is recommended that you set a lower amount of failed login attempts in clustered environments than in comparable non-clustered environments, since an attacker can distribute attacks over all nodes. You set the amount in the `csa.login.maxFailedAttempts` property described below.

Configure the Account Lockout Mechanism in the csa.properties File

To configure the account lockout mechanism, complete the following steps:

1. Open the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEBINF\classes\csa.properties` file in a text editor.

2. Locate the `#Enable the account lockout mechanism entry`.
3. Change one or more of the following properties as needed:

Property	Description
csa.login.lockout.enable	Required to enable the account lockout mechanism. To disable, set this property to false. It may be useful to disable account lockout in the case where an attacker continues to lock system accounts to cause denial of service, and the administrator is confident that all Codar Console users have very strong, secret passwords. **Default**: true
csa.login.maxFailedAttempts	The amount of failed login attempts that will lock the account. **Note:** It is recommended that you set a lower amount of failed login attempts in clustered environments than in comparable non-clustered environments, since an attacker can distribute attacks over all nodes. **Default**: true
csa.login.watchSeconds	The length of the watch period since the last failed login attempt after which the counter of failed login attempts will be reset. **Default**: 60 seconds
csa.login.lockSeconds	The length of the lockout period after which the account will be allowed to log in again. **Default**: 300 seconds (5 minutes)

4. Save and exit the file.
5. Restart the Codar service. See "Restart Codar" on page 60for instructions.

Configure IPv6

This chapter explains how to configure Codar to support IPv6 (both dual-stack and IPv6-only). Make sure that IPv6 has been implemented on the system on which Codar is running (including configuring the network and DNS) and that your web browser, such as Firefox or Chrome, have been enabled for IPv6 support.

To configure Codar to support IPv6, open `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` in a text editor and make the following changes:

1. Locate the following line:

   ```
   <wsdl-host>${jboss.bind.address:127.0.0.1}</wsdl-host>
   ```

2. Replace `127.0.0.1` with `[::1]`. For example, `<wsdl-host>${jboss.bind.address:[::1]}</wsdl-host>`
3. Locate the following lines:

   ```
   <interface name="management">
   <inet-address value="127.0.0.1" />
   </interface>
   ```

4. Replace `127.0.0.1` with `[::1]`. For example,

   ```
   <interface name="management">
   <inet-address value="[::1]" />
   </interface>
   ```

5. Locatethe following lines:

   ```
   <interface name="public">
   <inet-address value="0.0.0.0" />
   </interface>
   ```

6. Replace `0.0.0.0` with `[::]`. For example,

   ```
   <interface name="public">
   <inet-address value="[::]" />
   </interface>
   ```

7. Locate the following lines:

   ```
   <interface name="unsecure">
   <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}" />
   </interface>
   ```

8. Replace `127.0.0.1` with `[::1]`. For example,

   ```
   <interface name="public">
   ```

```
    <inet-address value="${jboss.bind.address.unsecure:[::1]}" />

    </interface>
```

To configure Codar tools (such as the process definition tool, purge tool, schema installation tool, provider tool, or content archive tool) to support IPv6, do the following:

When you configure the `db.url`, `dbUrl`, or `jdbc.databaseUrl` attribute in the database file used by the tool (for example, `config.properties`, `jdbc.properties`, or `db.properties`), enclose the IPv6 address in square brackets (for example, `[f000:253c::9c10:b4b4]` or `[::1]`).

Launch the Codar Console

Launch the Codar Console using an IPv6 address by typing the following URL in a supported web browser: `https://<ipv6_address>:8444/csa/login`

Common Access Card

This chapter provides information about the integration between a Common Access Card (CAC) and Codar, where Common Access Card is used as the user authentication mechanism. By configuring Common Access Card, you are able to log into Codar using a Personal Identity Verification card.

After integrating Codar with Common Access Card, the following log in rules apply:

- You can log in to the Codar Console using a Personal Identity Verification card with a valid certificate.
- You can log in to the Codar Console using an Codar out-of-the-box user account without a Personal Identity Verification card.
- You can only log in to the Codar Console as a valid LDAP user, **with** a Personal Identity Verification card.

Caution: For the Codar Console, single sign-on (SSO) cannot be enabled at the same time as Common Access Card.

Complete the following steps to integrate Codar with Common Access Card:

- Stop Codar
- "Update JBoss configuration to set up client authentication" below
- "Configure Codar Console" on page 78
- "Configure certificate revocation" on page 79
- "Start Codar" on page 81

Stop Codar

If Codar is running, stop Codar. See "Stop Codar" on page 59 for instructions.

Update JBoss configuration to set up client authentication

To update the JBoss configuration, complete the following steps:

1. Download the CA certificate for the digital certificate from the Personal Identity Verification card.
2. Import the CA certificate into a new truststore.

   Windows:

   The truststore type is determined by the Codar environment. That is, if Codar is running in a standard environment, the truststore type must be JKS.

   For example, in a standard environment, if you named the CA certificate from step 1 `CACcert.cer`, saved it in C:\ and wanted to create a truststore named `CSA_HOME\jboss-as\standalone\configuration\.piv_keystore`, run the following command:

   ```
   "CSA_JRE_HOME\bin\keytool" -importcert -file C:\CACcert.cer -alias caccert -keystore CSA_HOME\jboss-as\standalone\configuration\.piv_keystore -storepass <password>
   ```

   Linux:

The truststore type must be JKS.

For example, if you named the CA certificate from step 1 `CACcert.cer`, saved it in `/tmp`, and wanted to create a truststore named `CSA_HOME/jboss-as/standalone/configuration/.piv_keystore`, run the following command:

```
CSA_JRE_HOME/bin/keytool -importcert -file /tmp/CACcert.cer -alias caccert -keystore
CSA_HOME/jboss-as/standalone/configuration/.piv_keystore -storepass <password>
```

3. Edit the `CSA_HOME\[[[Undefined variable CSAVariables.dirjboss]]]\standalone\configuration\standalone.xml` file:
   a. Locate the `<security-realm name="CsaRealm">` element. Within this element and after `</server-identities>`, add the following:

```
<authentication>
   <truststore path="<location of truststore>" keystore-password="<truststore password>"/>
</authentication>
```

For example,

```
<security-realm name="CsaRealm">
   <server-identities>
      <ssl>
         <keystore keystore-password="changeit" path="C:\Program Files\HPE\CSA/[[[Undefined
variable CSAVariables.dirjboss]]]/standalone/configuration/.keystore"/>
      </ssl>
   </server-identities>
   <authentication>
      <truststore path="C:\Program Files\HPE\CSA\[[[Undefined variable
CSAVariables.dirjboss]]]\
standalone\configuration\.piv_keystore" keystore-password="TruststorePassword"/>
   </authentication>
</security-realm>
```

Linux:

```
<security-realm name="CsaRealm">
   <server-identities>
      <ssl>
         <keystore keystore-password="changeit" path="/usr/local/hpe/[[[Undefined variable
CSAVariables.dirjboss]]]/standalone/configuration/.keystore"/>
      </ssl>
   </server-identities>
   <authentication>
      <truststore path="/usr/local/hpe/[[[Undefined variable CSAVariables.dirjboss]]]
/standalone/configuration/.piv_keystore" keystore-password="TruststorePassword"/>
   </authentication>
</security-realm>
```

> **Note:** This example stores the password in clear text. If you want to use an encrypted password, see "Masking Passwords in standalone.xml Using the JBoss vault Script" on page 29 for information about creating a password vault for JBoss.

   b. Locate the `https-listener` element that contains the `name="https` and `security-realm="CsaRealm"` attributes. Add the `verify-client="REQUESTED"` attribute to this element.

For example,

```
<https-listener enabled-cipher-suites="TLS_ECDHE_ECDSA_WITH_AES_256_CBC_
SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, ... " name="https" security-
realm="CsaRealm" socket-binding="https" verify-client="REQUESTED"/>
```

Configure Codar Console

Complete the following steps to integrate the Codar Console with the Common Access Card:

1. Open the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\csa.properties` file in a text editor and uncomment the following line:

   ```
   enableCAC=true
   ```

2. Extract the user name from the certificate using the username extraction mechanism.

   The username extraction mechanism depends on the format of your certificate. The user name extracted from the certificate should match the user names configured in the LDAP configuration configured in CSA. CSA enables you to extract the user name using the **SubjectDN** and **Subject Alternative Name** (**SAN**) mechanisms. To configure the username extraction mechanism you must make the changes to the following properties in the `csa.properties` file:

Property	Description
csa.cac.x509Attribute	The name of the `X.509` certificate attribute from which the user name will be extracted. Set this property to `subjectDN/san/subjectDN`,san. If this property is set to contain both attributes such as `subjectDN,san` or `san,subjectDN`, then username will be extracted from the `subjectDN` attribute only if the SAN attribute is not present in the certificate. If this property is not set, then the default value for the property is "`subjectDN`".
csa.cac.regex	The regular expression used to extract a user name from the `subjectDN X.509` attribute. If this property is not set, then the default for regex is `CN=(.*?)`. This property need not be set if the property `csa.cac.x509Attribute` is set to "`san`".
csa.cac.san.type	The type of the subject alternative name. The allowed types are `othername` and `rfc822name`. If this property is not set, then the default value for the property is `otherName`. This property need not be set if `csa.cac.x509Attribute` is set to "`subjectDN`".

3. Navigate to the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\` directory.
4. Make a backup copy of the `applicationContext-security.xml` file.
5. Update the Spring Security configuration. Open the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml` file in a text editor and make the following changes:
   a. Locate the comment "Pre-authentication for CAC" and uncomment the following line:

      ```
      <security:authentication-provider ref="customX509AttrPreAuthAuthProvider"/>
      ```

   b. Locate and uncomment both occurrences the following line:

      ```
      <custom-filter position="LAST" ref="cacFilter" />
      ```

      > **Note:** The `<custom-filter position="LAST" ref="cacFilter" />` line defines the custom filter to be used and specifies that it will need to be set as the LAST filter in the chain of filters.

   c. Locate and uncomment both occurrences the following line:

```
<custom-filter position="X509_FILTER" ref="cacX509AuthenticationFilter" />
```

> **Note:** The URL must start with `http://` and cannot start with just `www`.

d. Locate the comment `Bean definitions for CAC` and uncomment the content that follows it:

```
<beans:bean id="cacUserDetailsService"
class="com.hp.csa.authn.impl.CACUserDetailsServiceImpl">
<beans:property name="restRole" value="ROLE_REST" />
</beans:bean>
<beans:bean id="cacFilter" class="com.hp.csa.security.CACFilter" />
<beans:bean id="cacX509AuthenticationFilter"
class="org.springframework.security.web.authentication.preauth.x509.X50
9AuthenticationFilter">
<beans:property name="authenticationManager"
ref="authenticationManager" />
<beans:property name="principalExtractor"
ref="customX509Extractor" />
</beans:bean>
<beans:bean id="customX509AttrPreAuthAuthProvider"
class="org.springframework.security.web.authentication.preauth.PreAuthe
nticatedAuthenticationProvider">
<beans:property name="preAuthenticatedUserDetailsService"
ref="customAuthenticationUserDetailsService" />
</beans:bean>
<beans:bean id="customAuthenticationUserDetailsService"
class="org.springframework.security.core.userdetails.UserDetailsByNameS
erviceWrapper">
<beans:property name="userDetailsService"
ref="cacUserDetailsService" />
</beans:bean>
<beans:bean id="customX509Extractor"
class="com.hp.csa.security.CustomX509PrincipalExtractor">
<beans:property name="x509Attribute"
value="${csa.cac.x509Attribute:subjectDN}"/>
<beans:property name="regex" value="${csa.cac.regex:CN=(.*?),}
"/>
<beans:property name="sanType"
value="${csa.cac.san.type:otherName}"/>
</beans:bean>
```

Configure certificate revocation

You will need to revoke a certificate if it has been compromised in any way or if an employee leaves your organization.

The following are the methods to revoke a certificate:

- "Configure Codar to use a Certificate Revocation List" on the next page
- "Configure Codar to use Certificate Revocation List Distribution Point" on the next page
- "ConfigureCodar to Use Online Certificate Status Protocol" on the next page

Configure Codar to use a Certificate Revocation List

The following is an example of how to revoke a certificate that was generated by the certificate authority and publish a Certificate Revocation List (CRL) that contains this certificate ID in the list. The CRL must already exist. You will download and save it in a folder on the system where Codar is installed and point to its location using the `ca-revocation-url` parameters.

1. Copy the CRL file to the system where Codar is installed (for example, copy it to the *<crl_file_directory>* directory).
2. In the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file, add the `ca-revocation-url="`*<crl_file_directory>*`"` attribute to the `<ssl>` element.

   For example, change the following from:

   ```
   <ssl name="ssl" key-alias="CSA" certificate-key-file="CSA_HOME\
   jboss-as\standalone\configuration\.keystore"
   ca-certificate-file="CSA_JRE_HOME\lib\security\cacerts"
   verify-client="want"/>
   ```

   to

   ```
   <ssl name="ssl" key-alias="CSA" certificate-key-file="CSA_HOME\
   jboss-as\standalone\configuration\.keystore"
   ca-certificate-file="CSA_JRE_HOME\lib\security\cacerts"
   verify-client="want" ca-revocation-url="<crl_file_directory>" />
   ```
3. Restart Codar service, see "Restart Codar" on page 60.
4. Log in to the Codar Console using a revoked certificate. The `Secure Connection Failed` message should display in the browser.

Configure Codar to use Certificate Revocation List Distribution Point

To enable a Certificate Revocation List Distribution Point (CRL DP), do the following:

1. Edit the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file and enable revocation and CRL DP by adding the following lines under `<system-properties>`:

   ```
   <property name="com.sun.net.ssl.checkRevocation" value="true"/>
   <property name="com.sun.security.enableCRLDP" value="true"/>
   ```
2. Restart Codar service, see "Restart Codar" on page 60.

ConfigureCodar to Use Online Certificate Status Protocol

To enable the Online Certificate Status Protocol (OCSP), complete the following steps:

1. Edit the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file and enable revocation by adding the following line under `<system-properties>`:

   ```
   <property name="com.sun.net.ssl.checkRevocation" value="true"/>
   ```
2. Edit the `CSA_JRE_HOME\lib\security\java.security` file and uncomment the following line:

```
ocsp.enable=true
```

3. Restart Codar service, see "Restart Codar" on page 60.

Start Codar

See "Start Codar" on page 59 for instructions.

Single Sign-On

This chapter provides information about integrating Codar with a single sign-on solution.

Tasks include:

- "Integrate with Single Sign-On" below
- "Integrate Codar with single sign-on solution" on page 85
- "Integrate Codar with CA SiteMinder" on page 87

Integrate with Single Sign-On

Single Sign-On is included with Codar and can be used only from the Codar Console when launching an application from the Codar Console. Single Sign-On must be installed and configured on the application before single sign-on can be integrated between it and Codar.

Details on how to integrate Single Sign-On between Codar and Operations Orchestration are included in the documentation for Codar. Information regarding Operations Orchestration can be found in this guide (the tasks are located in "Operations Orchestration" on page 40).

If you want to integrate Single Sign-On between Codar and another application (the application must be launched from the Codar Console), you must use Codar's `crypto InitString` attribute value. This value can be found in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\hpssoConfiguration.xml` file. Information on how to integrate Single Sign-On between Codar and other applications is not provided in this guide.

The following sections describe how to enable Single Sign-On if it was not enabled during installation and how to disable Single Sign-On.

Enable Single Sign-On

Codar installs Single Sign-On during installation which may have been enabled or disabled. If Single Sign-On was not enabled during installation and you want to start using Single Sign-On, complete the following tasks:

> **Note:** If you enabled Single Sign-On during the installation of Codar, you do not need to complete these tasks.

> **Caution:** If Single Sign-On and CA SiteMinder are both configured for Codar, and if only Single Sign-On is enabled for another application, a user logging out from the other application will not be logged out from Codar. For example, if Single Sign-On is enabled between Codar and Operations Orchestration, when a user logs out from Operations Orchestration Central, the user will not be logged out from the Codar Console.

"Step 1: Configure the domain" on the next page

"Step 2: Set the Single Sign-On property" on the next page

"Step 3: Configure the Identity Management component" on the next page

"Step 4: Restart Codar" on page 84

Step 1: Configure the domain

Configure the domain name of the network of the server on which Codar is installed. Applications launched from the Codar Console with which you want to use Single Sign-On must be installed on systems that belong to this domain.

To configure the domain, complete the following steps:

1. Navigate to the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF` directory.
2. Make a backup copy of the `hpssoConfiguration.xml` file.
3. Open the `hpssoConfiguration.xml` file in a text editor.
4. Locate the following content:

   ```
   <creationDomains>
      <domain>sso.domain</domain>
   </creationDomains>
   ```

5. Change `sso.domain` to domain name of the network of the server on which Codar is installed. Applications launched from the Codar Console with which you want to use Single Sign-On must be installed on systems that belong to this domain.

   For example, if your system host name is `codar_system.xyz.com`, enter `xyz.com` as the domain name.

6. Save and exit the file.

Step 2: Set the Single Sign-On property

To set the Single Sign-On property, complete the following steps:

1. Navigate to the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes` directory.
2. Make a backup copy of the `csa.properties` file.
3. Open the `csa.properties` file in a text editor.
4. Locate the following content:

   ```
   #enableHPSSO=true
   ```

5. Uncomment this line.
6. Save and exit the file.
7. Optionally, change the value of the `initString` setting for the Codar Console. If you create a new string, HPE recommends using at least 44 characters that are made up of ASCII letters, numbers, and basic symbols (ones that do not need to be escaped). The `initString` value represents a secret key and must be treated as such in your environment (this string is used to encrypt and decrypt the LWSSO_COOKIE_KEY cookie that is used to authenticate the user for single sign-on).
   a. Navigate to the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF` directory.
   b. Make a backup copy of the `hpssoConfiguration.xml` file and open it in an editor.
   c. Locate the `crypto` element and replace the value of `initString`.
   d. Save and exit the file.

Step 3: Configure the Identity Management component

To configure the Identity Management component, complete the following steps:

1. Navigate to the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF` directory.
2. Open the `web.xml` file in a text editor.
3. Locate the following comment (near the end of the file):

   ```
   <!-- START HP SSO Configuration -->
   ```
4. Uncomment the following content after this comment:

   ```
   <listener>
      <listener-class>com.hp.ccue.identity.hpssoImpl.HpSsoContextListener</listener-
   class>
   </listener>

   <context-param>
      <param-name>com.hp.sw.bto.ast.security.lwsso.conf.fileLocation</param-name>
      <param-value>C:\Program Files\HPE\Codar\jboss-as-7.1.1.Final\
   standalone\deployments\idm-service.war\WEB-INF\hpssoConfig.xml</param-value>
   </context-param>
   ```
5. Update the directory path name in `<param-value>` from "jboss-as-7.1.1.Final" to "jboss-as." For example, change

   ```
   CSA_HOME\jboss-as-7.1.1.Final\
   standalone\deployments\idm-service.war\WEB-INF\hpssoConfig.xml</param-value>
   ```

   to

   ```
   CSA_HOME\jboss-as\standalone\
   deployments\idm-service.war\WEB-INF\hpssoConfig.xml</param-value>.
   ```
6. Save and exit the file.

Step 4: Restart Codar

See "Restart Codar" on page 60 for instructions.

Disable Single Sign-On

If you no longer want to use Single Sign-On, you can disable it.

To disable Single Sign-On, complete the following steps:

1. Navigate to the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes` directory.
2. Make a backup copy of the `csa.properties` file.
3. Open the `csa.properties` file in a text editor.
4. Locate the following content:

   ```
   enableHPSSO=true
   ```
5. Change **true** to **false**.
6. Save and exit the file.
7. Restart Codar, see "Restart Codar" on page 60.

Integrate Codar with single sign-on solution

While Codar provides a single-sign-on solution using CA SiteMinder, there are a variety of scenarios where you may need to perform the integration with Codar using single-sign-on solution. For example, you may be using:

- An implementation where you need to authenticate with a single-sign-on vendor other than CA SiteMinder.
- A different deployment architecture than what is provided by Codar.
- A different version of CA SiteMinder than what is supported by Codar.
- An entirely different architecture than that which is supported.

In such cases it makes sense to create a custom single-sign-on solution so that you can extend the HPE-provided implementation to your own.

For the Codar Console, single-sign-on cannot be enabled at the same time as Common Access Card.

The following sections describe how to integrate Codar with a single sign-on solution.

- "Verify Codar provider organization's LDAP server configuration" below
- "Verify Codar consumer organization's LDAP server configuration" on the next page
- "Configure custom single-sign-on server to work with Codar" on the next page
- "Stop Codar" on the next page
- "Configure Codar Console" on the next page
- "Configure proxy mapping" on the next page
- "Start Codar" on page 87
- "Verify single-sign-on integration" on page 87

Verify Codar provider organization's LDAP server configuration

You should verify that an LDAP user can log into the Codar Console and the Marketplace Portal, which should already be configured. By performing this verification, you can be confident that any login issues that occur after integration have nothing to do with this particular configuration.

If there are any login issues, then update or configure the LDAP server for both the provider organization and the consumer organization from the Codar Console, which is the interface from which you perform all administration tasks for *both* the Codar Console and the Marketplace Portal.

> Note: You must configure the Codar Provider organization to use the same LDAP server used by the custom single sign-on server. If you do not configure this access point, no one will be able to access the Codar Console.

To configure or update the provider organization's LDAP server, complete the following steps:

1. Launch the Codar Console by typing the following URL in a supported web browser: `https://<codarhostname>:8444/csa` where *`<codarhostname>`* is the fully-qualified domain name of the system on which the Codar Console resides.
2. Log in to the Codar Console as a Codar Administrator.
3. Click the **Organizations** tile.

4. In the left-navigation frame, select the provider organization.
5. From the provider organization's navigation frame, select **LDAP**.
6. Update the LDAP server information.
7. Click **Save**.

Verify Codar consumer organization's LDAP server configuration

> **Note:** The same LDAP server must be used by the Codar Provider organization, Codar consumer organization and custom single sign-on server.

To configure or update the consumer organization's LDAP server, complete the following steps:

1. Launch the Codar Console by typing the following URL in a supported web browser: `https://<codarhostname>:8444/csa` where *`<codarhostname>`* is the fully-qualified domain name of the system on which the Codar Console resides.
2. Log in to the Codar Console as the Codar Administrator.
3. Click the **Organizations** tile.
4. In the left-navigation frame, select a consumer organization.
5. From the consumer organization's navigation frame, select **LDAP**.
6. Update the LDAP server information.
7. Click **Save**.
8. Repeat these steps for every consumer organization configured in Codar.

Only the `/codar` context is supported (this is required by the single sign-on proxy setup).

Configure custom single-sign-on server to work with Codar

To configure your custom single-sign-on server to work with Codar, follow the instructions provided with your single-sign-on application.

Stop Codar

See "Stop Codar" on page 59 for instructions.

Configure Codar Console

To configure the Codar Console, complete the following steps:

1. Update the `applicationContext-security.xml` file as appropriate for your custom single sign-on solution (based on the Spring Security Framework documentation).
2. Update the `csa.properties` file by uncommenting the string `enableSSO=true` and setting the value of `csa.subscriber.portal.url` to `{<protocol>}://{<host>}/mpp/org/{<orgName>}`.

Configure proxy mapping

To configure proxy mapping, complete the following steps:

1. Map the `/codar` proxy to the Codar deployment.

   > **Caution:** Use only `/codar` as the alias. Using another alias may cause Codar to fail.
   >
   > For example, when configuring the alias in an Apache proxy server, set the following:
   >
   > ```
   > ProxyPass /codar/ https://<codarhostname>:8444/csa/
   > ProxyPassReverse /codar/ https://<codarhostname>:8444/csa/
   > ```

2. Map the `/idm-service` proxy to the identity management (IdM) deployment.

Start Codar

See "Start Codar" on page 59 for instructions.

Verify single-sign-on integration

You should verify that the single-sign-on integration works by logging into the Codar Console using the newly-integrated single-sign-on solution.

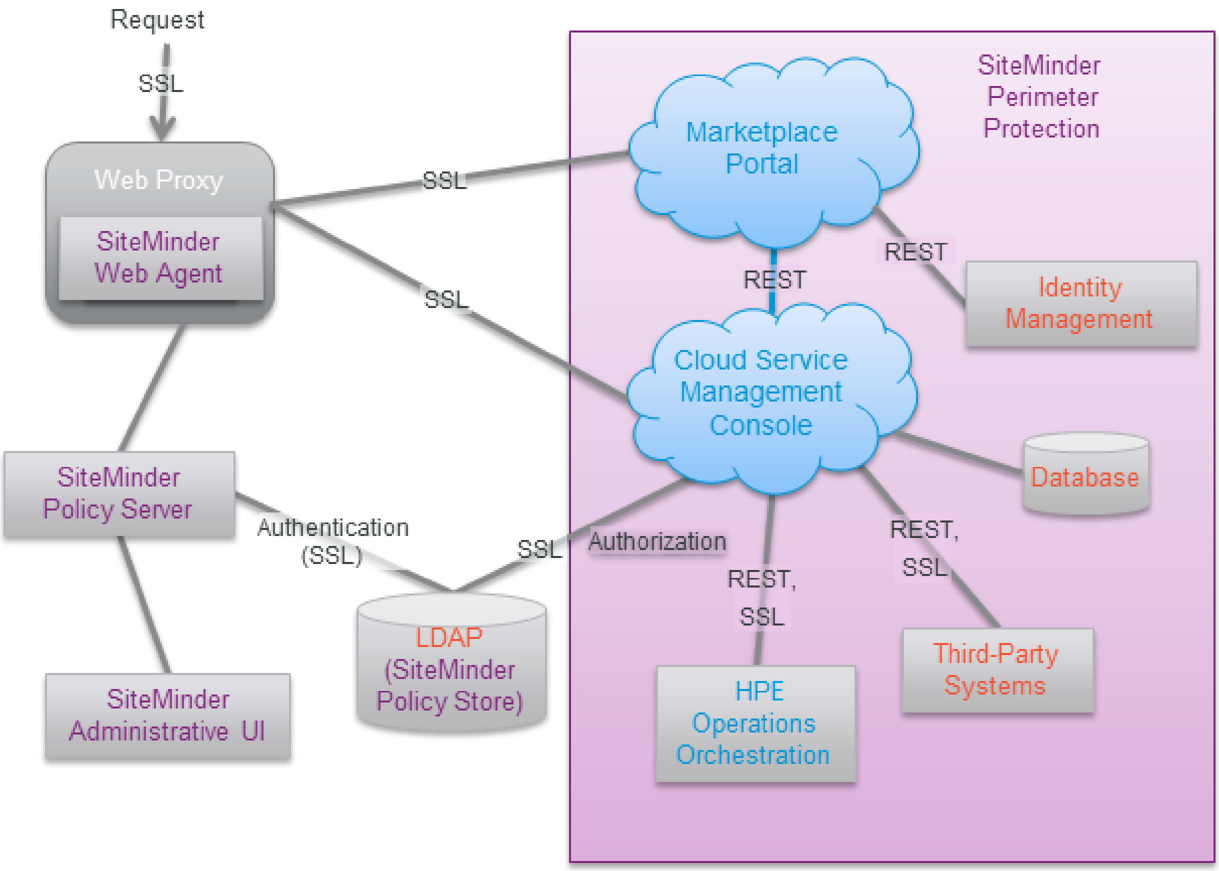
Integrate Codar with CA SiteMinder

Codar, as well as SiteMinder (also called CA Single Sign-On) with a reverse proxy solution, must already be installed and configured before you can integrate them. The LDAP server shared by Codar and SiteMinder must be configured for the Codar provider and consumer organization (from the Codar Console) before integration between Codar and SiteMinder is started.

SiteMinder is made up of several components that work with Codar and your LDAP server to provide secure access. The information provided in this section configures Codar to work with a reverse proxy solution, as shown in the following diagram.

> **Note:** The Marketplace Portal will only be available if you have both Cloud Service Automation and Codar licenses. For details on the Marketplace Portal, see the *Cloud Service Automation Configuration Guide*.

Supported SiteMinder Deployment Architecture



For more information about how to install and configure CA SiteMinder for a reverse proxy solution, see the *Configure Reverse Proxy Servers* section in the *Web Agent Configuration Guide* (a Web Agent guide). Documentation for SiteMinder can be found using the following URL:

https://support.ca.com/irj/portal/anonymous/DocumentationSearch

The following sections describe how to integrate Codar and SiteMinder:

- "Configure Codar provider organization's LDAP server"
- "Configure SiteMinder Policy Server for Codar integration"
- "Configure Codar for SiteMinder integration"
- "Customize Logout page (optional)"

Configure Codar provider organization's LDAP server

You must configure the Codar provider organization to use the same LDAP server used by the SiteMinder Policy Server. If you do not configure this access point before integrating Codar and SiteMinder, you will not be able to access Codar after integration.

> **Caution:** LDAP must be configured for the Codar provider organization before you begin the integration between Codar and SiteMinder. After integrating Codar and SiteMinder, you can only log in to the Codar Console via SiteMinder using a valid user from this LDAP directory. The out-of-the-box Codar users can no longer be used to log in to Codar.
>
> When using the REST API, the out-of-the-box Codar users are still valid after integration.

To configure the provider organization's LDAP server, do the following:

1. Launch the Codar Console by typing the following URL in a supported web browser: `https://<codarhostname>:8444/csa` where *`<codarhostname>`* is the fully-qualified domain name of the system on which the Codar Console resides.
2. Log in to the Codar Console as a Codar Administrator.
3. Click the **Organizations** tile.
4. In the left-navigation frame, select the provider organization.
5. From the provider organization's navigation frame, select **LDAP**.
6. Update the LDAP server information.
7. Click **Save**.

Configure SiteMinder Policy Server for Codar integration

To configure the SiteMinder Policy Server for Codar integration, complete the following steps:

1. Navigate to **Start** > **Administrative Tools** > **Services**.
2. Configure the SiteMinder Policy Server to use the LDAP server that will be shared between Codar and SiteMinder.
3. Configure the SiteMinder Policy Server idle timeout and the Codar Console session timeout, to be the same amount of time, regardless of the units (minutes or seconds) used by the parameters in the respective configuration files. By default, the session timeout value for the Codar Console is 60 minutes.

   The session timeout for the Codar Console is configured using the `session-timeout` parameter in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\web.xml` file:

   ```
   ...
   <session-config>
   ...
     <session-timeout>60</session-timeout>
   ...
   ```
4. To process image file names that contain spaces, from the SiteMinder Policy Server, either comment out the `BadUrlChars` parameter or modify the SiteMinder Policy Server to allow image file names that contain spaces.
5. If you have both Cloud Service Automation and Codar licenses, do the following:
   a. Navigate to **Start** > **Administrative Tools** > **Services**.
   b. Right-click on the **HPE Marketplace Portal service** and select **Start**.

Configure the SiteMinder Web Agent for Codar integration

Configure proxy mapping for the SiteMinder Web Agent. To configure proxy mapping:

1. Map the `/codar` proxy to the Codar deployment. Use only `/codar` as the alias. Using another alias may cause Codar to fail.

   For example:

   ```
   ProxyPass /codar/ https://<codarhostname>:8444/csa/
   ```

   ```
   ProxyPassReverse /codar/ https://<codarhostname>:8444/csa/
   ```

2. Map the `/idm-service` proxy to the Identity Management component deployment. For example:

   ```
   ProxyPass /idm-service/ https://<codarhostname>:8444/idm-service/
   ```

   ```
   ProxyPassReverse /idm-service/ https://<codarhostname>:8444/idm-service/
   ```

Configure Codar for SiteMinder integration

To configure Codar for SiteMinder integration, you must do the following:

- "Stop Codar" below
- "Configure Codar Console" below
- "Start Codar" on the next page

Stop Codar

See "Stop Codar" on page 59for instructions.

Configure Codar Console

Configure the Codar Console for a SiteMinder reverse proxy solution. Update the `applicationContext-security.xml` file.

To configure Codar Console, complete the following steps:

1. Navigate to the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF` directory.
2. Make a backup copy of the `applicationContext-security.xml` file.
3. Open the `applicationContext-security.xml` file in a text editor.
4. Locate the `SSO Authentication Provider` comment and uncomment the following content that appears after this comment:

   ```
   <security:authentication-provider ref='ssoAuthenticationProvider' />
   ```

5. Locate the `custom filter config for SSO` comment and uncomment the following content that appears after this comment:

   ```
   <custom-filter position="PRE_AUTH_FILTER" ref="ssoSiteminderFilter" />
   ```

6. Locate the `Below is logout filter definition` comment and uncomment the following content that appears after this comment:

   ```
   <beans:constructor-arg value="/ssologout.jsp"/>
   ```

7. In the same section of the file, comment out the following content:

   ```
   <beans:constructor-arg value="/logout.jsp"/>
   ```

8. Locate the `Bean definitions for SSO` comment and uncomment the following content that appears after this comment:

   ```
   <beans:bean id="ssoSiteminderFilter"
    class="com.hp.csa.authn.impl.SSOHeaderAutheticationFilter">
      <beans:property name="principalRequestHeader" value="SM_USER" />
      <beans:property name="authenticationManager"
       ref="authenticationManager" />
      <beans:property name="exceptionIfHeaderMissing" value="true" />
      <beans:property name="ignoreURLContaining">
         <beans:list>
   ```

```
         <beans:value>/csa/rest/</beans:value>
         <beans:value>/csa/api/blobstore</beans:value>
      </beans:list>
   </beans:property>
</beans:bean>

<beans:bean id="ssoAuthenticationProvider"
 class="org.springframework.security.web.authentication.preauth.
 PreAuthenticatedAuthenticationProvider">
   <beans:property name="preAuthenticatedUserDetailsService">
      <beans:bean id="userDetailsServiceWrapper"
       class="org.springframework.security.core.userdetails.
       UserDetailsByNameServiceWrapper">
         <beans:property name="userDetailsService"
          ref="ssoPreAuthenticatedUserDetailsService" />
      </beans:bean>
   </beans:property>
</beans:bean>
<beans:bean id="ssoPreAuthenticatedUserDetailsService"
class="com.hp.csa.authn.impl.SSOUserDetailsService">
   <beans:property name="restRole" value="ROLE_REST" />
</beans:bean>
```

9. Save and exit the file.
10. Navigate to the `classes` subdirectory, `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes`.
11. Open the `csa.properties` file in a text editor.
12. Edit the following line to configure the URL to display for the organization in the Codar Console:

    `codar.subscriber.portal.url={protocol}://{host}:8089/org/{orgName}`

    You can define a hard-coded URL or a URL that is replaced by information as known by the client-side browser. The following tokens are supported: `protocol` (`http` or `https`), `host` (the host in the browser URL used to access the Codar Console), and `orgName` (the organization name of the selected organization in the browser). For example, if the client URL is `https://codar-server.company.com:8444/csa`, for a selected organization named `devteam`, then after the token replacement, the client displays a URL of `https://codar-server.company.com:8089/#/login/devteam`. No port is defined.
13. Locate the `Needed for SSO` comment and uncomment the following content:

    `enableSSO=true`
14. Save and exit the file.

Start Codar

See "Start Codar" on page 59 for instructions.

Customize Logout page (optional)

After clicking the `Log out` link from the Codar Console, the user is directed to a logout page. This page is customizable.

The following is the name and location of the logout file. There is one file for the Codar Console.

- Codar Console:

  ```
  CSA_HOME\jboss-as\standalone\deployments\csa.war\ssologout.jsp
  ```

> **Note:** By default, after logging out, the user must close the web browser in order to completely clear the SiteMinder session.
> The logout page can be customized to point to a SiteMinder logout page if one is available.

Database administration

This chapter provides miscellaneous information about maintaining the database.

Tasks include:

- "Restart database and Codar service" below
- "Configure Codar reporting database user" below
- "Update Codar database user or password" on page 96 (required if you change the database user or password)
- "Import large archives" on page 97
- "Install Codar database schema" on page 98
- "Configure Codar to mitigate frequently dropped database connections" on page 104

Restart database and Codar service

If you restart the database, you must restart the Codar service. If you do not restart the service, you may not be able to log in to the Codar Console.

Restart Codar service on Windows

To restart the Codar service on Windows, complete the following steps:

1. On the server that hostsP Codar, navigate to **Start** > **Administrative Tools** > **Services**.
2. Right-click on the Codar service and select **Restart**.

Restart Codar service on Linux

To restart the Codar service on Linux:

On the server that hosts Codar, type the following:

```
service codar start
```

Configure Codar reporting database user

This section explains how to configure the Codar reporting database user and role and run the schema installation script to define a read-only user required to use the reporting capabilities of Codar.

If you already configured the Codar reporting database user and role and defined the Codar reporting database user when running the installer or upgrade installer, you do not need to repeat these steps (the Codar reporting database user is already configured).

If you installed or upgraded Codar but did not configure the Codar reporting database user during the installation or upgrade and want to use the reporting capabilities of Codar, complete the tasks in this section.

To configure the Codar reporting database user, complete the following steps:

1. Create a read-only user.

   > **Caution:** The user name cannot contain more than one dollar sign symbol (`$`). For example, `c$adb` is a valid name but `c$$adb` and `c$ad$b` are not valid names.

   For example, do one of the following, based on the database you are using with Codar:

   Oracle

   Run the following commands to create the `CodarReportingDBRole` role and `CodarReportingDBUser` user:

   ```
   Create user CodarReportingDBUser identified by CodarReportingDBUser;
   Create role CodarReportingDBRole;
   Grant CREATE SESSION to CodarReportingDBUser;
   Grant CodarReportingDBRole to CodarReportingDBUser;
   Alter user CodarReportingDBUser default role CodarReportingDBRole;
   ```

   You will also need to add the CREATE ANY SYNONYM privilege to the Codar database user. This allows the Codar database user to create synonyms for the Codar reporting (read-only) database user.

   For example, if the Codar database user is named CodarDBUser, run the following command:

   ```
   Grant CREATE ANY SYNONYM to CodarDBUser
   ```

   Microsoft SQL

   Add a reporting database user (`CodarReportingDBUser`) to the Codar database with no roles:

   ```
   CREATE LOGIN CodarReportingDBUser WITH PASSWORD = '<codarreportingdbuser_password>';

   CREATE USER CodarReportingDBUser FOR LOGIN CodarReportingDBUser WITH DEFAULT_SCHEMA
   = codar;
   ```

   PostgreSQL

   From the psql prompt, enter the following:

   ```
   CREATE ROLE CodarReportingDBUser LOGIN PASSWORD '<codarreportingdbuser_password>'
   NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT;
   GRANT CONNECT ON DATABASE codardb to CodarReportingDBUser;
   ```

2. Run the following script:

   Oracle

   ```
   CSA_HOME\scripts\reporting\oracle\grant-reporting-user.sql
   ```

   Microsoft SQL

   ```
   CSA_HOME\scripts\reporting\mssql\grant-reporting-user.sql
   ```

   PostgreSQL

   ```
   CSA_HOME\scripts\reporting\postgresql\grant-reporting-user.sql
   ```

3. Restart Codar. See "Restart Codar" on page 60 for instructions.

4. The Codar reporting database user can access the data using the following view:

   ```
   RPT_RSC_CAPACITY_V
   ```

Update Codar database system

If you changed the hostname, domain, IP address, or port of the system on which the database used by Codar is installed, you must update the Codar configuration files that store this information.

1. If Codar is running, stop Codar. See "Stop Codar" on page 59.
2. On the system running Codar, open a command prompt and change to the `CSA_HOME\jboss-as\standalone\configuration` directory.
3. In a text editor, open the `standalone.xml` file.
4. In the file, locate the `<datasource>` element of the Codar database and the system information to be updated. For example:

   Microsoft SQL Server

   ```
   <datasource jndi-name="java:jboss/datasources/codarDS" pool-name="mssqlDS">
      <connection-
   url>jdbc:jtds:sqlserver://127.0.0.1:1433/codardb;ssl=request</connection-url>
      <driver>mssqlDriver</driver>
   .
   .
   .
   </datasource>
   ```

   Oracle

   ```
   <datasource jndi-name="java:jboss/datasources/codarDS" pool-name="OracleDS">
      <connection-url>jdbc:oracle:thin://127.0.0.1:1521/codardb</connection-url>
      <driver>oracleDriver</driver>
   .
   .
   .
   </datasource>
   ```

   PostgreSQL

   ```
   <datasource enabled="true" jndi-name="java:jboss/datasources/codarDS" jta="true"
   pool-name="codarPostgresDS" use-ccm="true" user-java-context="true">
      <connection-url>jdbc:postgresql://127.0.0.1:5432/codardb</connection-url>
      <driver>pgsqlDriver</driver>
   .
   .
   .
   </datasource>
   ```

5. The highlighted text should contain the old fully-qualified domain name, IP address, and/or port that must be updated. Replace this highlighted text with the new fully-qualified domain name, IP address, and/or port.
6. Save the `standalone.xml` file.
7. Restart Codar service, see "Restart Codar" on page 60.
8. If you are using a tool (such as the content archive tool, process definition tool, provider tool, purge tool, or schema installation tool) that uses a database or configuration properties file (for example,

`db.properties` or `config.properties`), update the appropriate property or properties in the file. By default, the file is located in the `CSA_HOME\Tools\<Tool_Name>` directory.

Update Codar database user or password

If you changed the user or password of the database used by Codar, you must update the JBoss DataSource and other files that store this information.

1. On the system running Codar, open a command prompt and change to the `CSA_HOME\jboss-as` directory.
2. Run the following command to generate an encoded version of the new database password:

   Windows:

   ```
   "CSA_JRE_HOME\bin\java" -cp "modules\org\jboss\logging\main\
   jboss-logging-3.1.2.GA.jar;modules\org\picketbox\main\
   picketbox-4.0.13.Final.jar"
   org.picketbox.datasource.security.SecureIdentityLoginModule <password>
   ```

   Linux:

   ```
   CSA_JRE_HOME/bin/java -cp "modules/org/jboss/logging/main/
   jboss-logging-3.1.2.GA.jar;modules/org/picketbox/main/
   picketbox-4.0.13.Final.jar"
   org.picketbox.datasource.security.SecureIdentityLoginModule <password>
   ```

   Copy the encoded password value that is returned (do not include spaces).
3. Stop the Codar service, see "Stop Codar" on page 59.
4. In a text editor, open the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file.
5. In the file, locate the following content:

   Microsoft SQL Server

   ```
   <security-domain name="codar-encryption-sec" cache-type="default">
      <authentication>
         <login-module
   code="org.picketbox.datasource.security.SecureIdentityLoginModule" flag="required">
            <module-option name="username" value="<old_user_name>"/>
            <module-option name="password" value="<old_encoded_password>"/>
            <module-option name="managedConnectionFactoryName"
   value="jboss.jca:service=LocalTxCM,name=mssqlDS"/>
         </login-module>
      </authentication>
   </security-domain>
   ```

   Oracle

   ```
   <security-domain name="codar-encryption-sec" cache-type="default">
      <authentication>
         <login-module
   code="org.picketbox.datasource.security.SecureIdentityLoginModule" flag="required">
            <module-option name="username" value="<old_user_name>"/>
            <module-option name="password" value="<old_encoded_password>"/>
            <module-option name="managedConnectionFactoryName"
   value="jboss.jca:service=LocalTxCM,name=OracleDS"/>
         </login-module>
   ```

```
   </authentication>
</security-domain>
```

PostgreSQL

```
<security-domain name="codar-encryption-sec" cache-type="default">
   <authentication>
      <login-module
code="org.picketbox.datasource.security.SecureIdentityLoginModule" flag="required">
         <module-option name="username" value="<old_user_name>"/>
         <module-option name="password" value="<old_encoded_password>"/>
         <module-option name="managedConnectionFactoryName"
value="jboss.jca:service=LocalTxCM,name=PostgresDS"/>
      </login-module>
   </authentication>
</security-domain>
```

6. Replace <*old_encoded_password*> with the new encoded password you copied in step 2 and <*old_user_name*> with the new user name.
7. Save the `standalone.xml` file.
8. Restart the Codar service, see "Restart Codar" on page 60.
9. If you are using a tool (such as the content archive tool, process definition tool, provider tool, purge tool, or schema installation tool) that uses a database or configuration properties file (for example, `db.properties` or `config.properties`), update the appropriate property or properties in the file. By default, the file is located in the `CSA_HOME\Tools\<Tool_Name>` directory.

   The password property value should be *encrypted* (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses.

Import large archives

Archives exported from Codar can be imported to install artifacts or update existing artifacts in Codar. Archives can be imported using the Codar Content Archive Tool, the Codar Console, or the REST API.

The default configuration for importing archives supports an archive up to 2 MB in size. When an archive larger than 2 MB is imported (typically, a catalog), the import operation may hang or take a very long time to complete. If an archive is larger than 2 MB, HPE recommends using the Content Archive Tool and increasing the JVM heap size.

Import large archives using Codar Content Archive Tool

If you want to import an archive larger than 2 MB, HPE recommends using the Content Archive Tool because the tool uses its own JVM heap (it does not share the JVM heap used by Codar). When you reconfigure the JVM heap size for the tool, you do not need to restart Codar and Codar performance is not affected by the import.

To increase the JVM heap size when running the Content Archive Tool, add the `-Xms<heap_size>M -Xmx<heap_size>M` options to the command line. For example, to increase the JVM heap size to 3 GB, type:

```
"CSA_JRE_HOME\bin\java -Xms3072M -Xmx3072M -jar content-archive-tool.jar -i -z catalog_archive.zip
```

> **Note:** By default, the JVM heap size used by the Content Archive Tool is 2 GB. If you want to use a larger JVM heap size, you must always specify the two options listed above when running the Content Archive Tool.

For more information about the Content Archive Tool, see the *Codar Content Archive Tool* guide.

Import large archives from Codar Console or through the REST API

If you want to import an archive larger than 1.5 MB, HPE recommends using the Content Archive Tool. If you must use the Codar Console or REST API to import a large archive, you must update the JVM heap size for Codar which requires Codar to be restarted. Also, importing a large archive from the Codar Console or through the REST API may slow the performance of Codar.

To increase the JVM heap size before importing a large archive from the Codar Console or through the REST API, complete the following steps:

1. If Codar is running, stop Codar. See "Stop Codar" on page 59.
2. Increase the JVM heap size for Codar.
   a. Open the `CSA_HOME\jboss-as\bin\standalone.conf.bat` file in a text editor.
   b. Locate the following line:
   ```
   set "JAVA_OPTS=%JAVA_OPTS%$JAVA_OPT -Xms2048M -Xmx2048M -
   XX:ReservedCodeCacheSize=256M
   -XX:MaxPermSize=256M"
   ```
   c. Increase the JVM heap size (by default, the JVM heap size is 1 GB). For example, to change the JVM heap size to 3 GB, change the line to:
   ```
   set "JAVA_OPTS=%JAVA_OPTS%$JAVA_OPT -Xms3072M -Xmx3072M -
   XX:ReservedCodeCacheSize=256M"
   ```
   d. Save and close the file.
3. Start Codar, see "Start Codar" on page 59.

For more information about importing archives from the Codar Console, see the Codar Console Help. For more information about importing archives through the REST API, see the *Codar API Reference* guide.

Install Codar database schema

The schema installation tool is used to upgrade the existing Codar database schema or install a fresh database schema without re-installing Codar. Use this tool if you did not install Codar database components onto the database during installation, did not upgrade the database schema during an upgrade, or if you want to drop the existing schema and install a fresh Codar database schema. You can also use this tool to complete an upgrade if the upgrade failed, the database schema was not updated, the failure was not due to a database problem, and the problem can be fixed without rerunning the upgrade installer. For example, if the upgrade failed but can be completed successfully by manual configuration but the database schema was not updated, you can simply make the manual changes to complete the upgrade and run the schema installation tool instead of reverting Codar back to the previous version and running the upgrade installer again.

> **Note:** Do not run this tool if you installed the database components during the installation of Codar or if you upgraded the database schema when you upgraded Codar.

If you run this tool on an existing schema (where Codar has been upgraded but the database schema was not upgraded), the schema is upgraded and no data in the database is lost. However, if you drop the existing schema and run this tool, all data in the database associated with the dropped schema is lost. Once you run the tool, a fresh schema is installed and you cannot revert back to the dropped schema.

Caution: Once you drop an existing schema and run the database schema installation tool, you cannot revert back to the dropped schema.

Upgrade or install database schema

To upgrade or install a fresh Codar database schema, complete the following steps:

1. If Codar is running, stop Codar. See "Stop Codar" on page 59.
2. Change to the `CSA_HOME\Tools\SchemaInstallationTool\` directory.
3. During upgrade or installation of Codar, a file named `db.properties` is generated in `CSA_HOME\Tools\SchemaInstallationTool\`. Verify the property values in this file. If you changed any database property values in the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file after installation, the values in `db.properties` may not be up-to-date.

   If you have dropped the existing database schema and are installing a fresh database schema after upgrading to Codar 1.60, you must update the `driverFiles` property value. The properties defined in `db.properties` are described in the following table.

Property Name	Description

dbScriptsDir	The location of database scripts installed with Codar used by the tool. If you are running a fresh installation of Codar 1.60 (you did not upgrade to Codar 1.60), you do not need to change these values.
	If you have upgraded to Codar 1.60 and want to upgrade the existing schema, you do not need to change these values.
	If you have upgraded to Codar 1.60, have dropped the existing database schema, and are installing a fresh database schema, you must update this value to the following:
	Oracle: (upgrade and dropped schema only)
	`dbScriptsDir=CSA_HOME\scripts\freshinstallscripts\oracle`
	PostgreSQL: (upgrade and dropped schema only)
	`dbScriptsDir=CSA_HOME\scripts\freshinstallscripts\postgresql`
	Microsoft SQL: (upgrade and dropped schema only)
	`dbScriptsDir=CSA_HOME\scripts\freshinstallscripts\mssql`

dbUrl	The JDBC URL. When specifying an IPv6 address, it must be enclosed in square brackets (see examples below). **Examples**

Property Name	Description

	Oracle (TLS not enabled):
	`jdbc.databaseUrl=jdbc:oracle:thin:@127.0.0.1:1521:XE`
	Oracle (TLS not enabled, using an IPv6 address):
	`jdbc.databaseUrl=jdbc:oracle:thin:@[f000:253c::9c10:b4b4]:1521:XE`
	Oracle (TLS enabled, Codar does not check the database DN):
	`jdbc.databaseUrl=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL = TCPS)(HOST = `*`<host>`*`)(PORT = 1521)))(CONNECT_DATA =(SERVICE_NAME = ORCL)))` where `<host>` is the name of the system on which the Oracle database server is installed.
	Oracle (TLS enabled, Codar checks the database DN):
	`jdbc.databaseUrl=jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCPS)(HOST = `*`<host>`*`)(PORT = 1521)))(CONNECT_DATA = (SERVICE_NAME = ORCL))(SECURITY=(SSL_SERVER_CERT_DN="CN=abc,OU=dbserver,O=xyz,L=Sunnyvale,ST=CA,C=US")))`
	where `<host>` is the name of the system on which the Oracle database server is installed and the values for `SSL_SERVER_CERT_DN` are for the DN of the Oracle database server.
	MS SQL (TLS not enabled):
	`jdbc.databaseUrl=jdbc:jtds:sqlserver://127.0.0.1:1433/example;ssl=request`
	MS SQL (TLS not enabled, using an IPv6 address):
	`jdbc.databaseUrl=jdbc:jtds:sqlserver://[::1]:1433/example;ssl=request`
	MS SQL (TLS enabled):
	`jdbc.databaseUrl=jdbc:jtds:sqlserver://127.0.0.1:1433/example;ssl=authenticate`
	PostgreSQL:
	`jdbc.databaseUrl=jdbc:postgresql://127.0.0.1:5432/codardb`

dbUserName	The user name of the database user you configured for Codar after installing the database.
dbPassword	The password for the database user. The password should be encrypted (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses. While you may enter a password in clear text, after you run the tool, the clear text password is automatically replaced by an encrypted password. **Example** `dbPassword=ENC(fc5e38d38a5703285441e7fe7010b0)`

Property Name	Description

driverFiles	The database driver files used by this tool.
	- You do not need to change these values if:
	  ○ You are running a fresh installation of Codar 1.60 (you did not upgrade to Codar 1.60).
	  ○ You upgraded to Codar 1.60 and want to upgrade the existing schema.
	- You must update this value to the value shown below, if you upgraded to Codar 1.60, dropped the existing database schema, and are installing a fresh database schema:
	Oracle (upgrade and dropped schema only)
	`driverFiles=CSA_HOME\scripts\schemainstallforupg\`
	`create-oracle-schema.sql,`
	`CSA_HOME\scripts\schemainstallforupg\`
	`create-oracle-topology-schema.sql,`
	`CSA_HOME\scripts\schemainstallforupg\oracle\`
	`seed_data_driver.sql,`
	`CSA_HOME\scripts\reporting\oracle\`
	`install_views_driver.sql,`
	`CSA_HOME\scripts\reporting\oracle\`
	`grant-reporting-user.sql`
	PostgreSQL (upgrade and dropped schema only)
	`driverFiles=CSA_HOME\scripts\schemainstallforupg\`
	`create-postgres-schema.sql,`
	`CSA_HOME\scripts\schemainstallforupg\`
	`create-postgres-topology-schema.sql,`
	`CSA_HOME\scripts\schemainstallforupg\postgres\`
	`seed_data_driver.sql,`
	`CSA_HOME\scripts\reporting\postgres\`
	`install_views_driver.sql,`
	`CSA_HOME\scripts\reporting\postgres\`
	`grant-reporting-user.sql`
	Microsoft SQL (upgrade and dropped schema only)
	`driverFiles=CSA_HOME/scripts/schemainstallforupg/`
	`alterdb.sql,`
	`CSA_HOME\scripts\schemainstallforupg\`
	`create-mssql-schema.sql,`
	`CSA_HOME\scripts\schemainstallforupg\`
	`create-mssql-topology-schema.sql,`
	`CSA_HOME\scripts\schemainstallforupg\`
	`mssql\seed_data_driver.sql,`
	`On Linux only:`
	`CSA_HOME\scripts\reporting\mssql\`
	`install_views_driver.sql,`
	`CSA_HOME\scripts\reporting\mssql\`
	`grant-reporting-user.sql`

Property Name	Description
	Note: Add the `grant-reporting-user.sql` file only if you have created the reporting database user for Codar.
jdbcDriverClassName	The JDBC driver class. Do not change this value. **Examples** **Oracle:** `jdbc.driverClassName=oracle.jdbc.driver.OracleDriver` **MS SQL:** `jdbc.driverClassName=net.sourceforge.jtds.jdbc.Driver` **PostgreSQL:** `jdbc.driverClassName=org.postgresql.Driver`
jdbcDriverDir	The location of the JDBC driver(s) used by this tool. Do not change this value.

4. Run the following command:

   Windows:

   - **Oracle (TLS not enabled), MS SQL, and PostgreSQL**

     ```
     "CSA_JRE_HOME\bin\java" -jar schema-installation-tool.jar
     ```

   - **Oracle (TLS enabled, Codar does not check the database DN, client authentication is enabled on the Oracle database server)**

     ```
     "CSA_JRE_HOME\bin\java" -Djavax.net.ssl.keyStore="<certificate_key_file>"
     -Djavax.net.ssl.keyStorePassword=<certificate_key_file_password>
     -Djavax.net.ssl.keyStoreType=<certificate_key_file_type>
     -jar schema-installation-tool.jar
     ```

     certificate_key_file is the same keystore file defined by the certificate-key-file attribute in the ssl element of the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file (for example, `CSA_HOME\jboss-as\standalone\configuration\.keystore`).

     certificate_key_file_password is the password to the keystore file.

     certificate_key_file_type is the keystore type (for example, JKS or PKCS12).

   - **Oracle (TLS enabled, Codar does not check the database DN, client authentication is NOT enabled on the Oracle database server)**

     ```
     "CSA_JRE_HOME\bin\java" -jar schema-installation-tool.jar
     ```

   - **Oracle (TLS enabled, Codar checks the database DN, client authentication is enabled on the Oracle database server)**

     ```
     "CSA_JRE_HOME\bin\java" -Doracle.net.ssl_server_dn_match=true
     -Djavax.net.ssl.keyStore="<certificate_key_file>"
     -Djavax.net.ssl.keyStorePassword=<certificate_key_file_password>
     -Djavax.net.ssl.keyStoreType=<certificate_key_file_type>
     -jar schema-installation-tool.jar
     ```

certificate_key_file is the same keystore file defined by the certificate-key-file attribute in the ssl element of the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file (for example, `CSA_HOME\jboss-as\standalone\configuration\.keystore`).

certificate_key_file_password is the password to the keystore file.

certificate_key_file_type is the keystore type (for example, JKS or PKCS12).

- **Oracle (TLS enabled, Codar checks the database DN, client authentication is NOT enabled on the Oracle database server)**

  ```
  "CSA_JRE_HOME\bin\java" -Doracle.net.ssl_server_dn_match=true
  -jar schema-installation-tool.jar
  ```

Linux:

- **Oracle (TLS not enabled), MS SQL, and PostgreSQL**

  ```
  CSA_JRE_HOME/bin/java -jar schema-installation-tool.jar
  ```

- **Oracle (TLS enabled, Codar does not check the database DN, client authentication is enabled on the Oracle database server)**

  ```
  CSA_JRE_HOME/bin/java -Djavax.net.ssl.keyStore="<certificate_key_file>"
  -Djavax.net.ssl.keyStorePassword=<certificate_key_file_password>
  -Djavax.net.ssl.keyStoreType=<certificate_key_file_type>
  -jar schema-installation-tool.jar
  ```

  certificate_key_file is the same keystore file defined by the certificate-key-file attribute in the ssl element of the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file (for example, `CSA_HOME\jboss-as\standalone\configuration\.keystore`).

  certificate_key_file_password is the password to the keystore file.

  certificate_key_file_type is the keystore type (for example, JKS or PKCS12).

- **Oracle (TLS enabled, Codar does not check the database DN, client authentication is NOT enabled on the Oracle database server)**

  ```
  CSA_JRE_HOME/bin/java -jar schema-installation-tool.jar
  ```

- **Oracle (TLS enabled, Codar checks the database DN, client authentication is enabled on the Oracle database server)**

  ```
  CSA_JRE_HOME/bin/java -Doracle.net.ssl_server_dn_match=true
  -Djavax.net.ssl.keyStore="<certificate_key_file>"
  -Djavax.net.ssl.keyStorePassword=<certificate_key_file_password>
  -Djavax.net.ssl.keyStoreType=<certificate_key_file_type>
  -jar schema-installation-tool.jar
  ```

  certificate_key_file is the same keystore file defined by the certificate-key-file attribute in the ssl element of the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file (for example, `CSA_HOME\jboss-as\standalone\configuration\.keystore`).

  certificate_key_file_password is the password to the keystore file.

  certificate_key_file_type is the keystore type (for example, JKS or PKCS12).

- **Oracle (TLS enabled, Codar checks the database DN, client authentication is NOT enabled on the Oracle database server)**
  ```
  CSA_JRE_HOME/bin/java -Doracle.net.ssl_server_dn_match=true
  -jar schema-installation-tool.jar
  ```

Configure Codar to mitigate frequently dropped database connections

If you are experiencing frequently dropped database connections, configure the JBoss data source connections to mitigate the problem.

In a standalone environment, complete the following steps:

1. Stop the Codar service, see "Stop Codar" on page 59.
2. Edit the `CSA_HOME\jboss-as\standalone\configuration\standalone.xml` file:
   a. Find the `dataSource` tag which is used for Codar database configuration.
   b. Add the following after the line that ends with `</security>`:

      Oracle:

      ```
      <validation>
      <check-valid-connection-sql>select 1 from DUAL</check-valid-connection-sql>
      <validate-on-match>false</validate-on-match>
      </validation>
      ```

      MS SQL or PostgreSQL:

      ```
      <validation>
      <check-valid-connection-sql>select 1</check-valid-connection-sql>
      <validate-on-match>false</validate-on-match>
      </validation>
      ```
3. Start the Codar service, see "Start Codar" on page 59.

In a clustered environment, complete the following steps:

1. Stop the Codar service, see "Stop Codar" on page 59.
2. Edit the `CSA_HOME\jboss-as\domain\configuration\domain.xml` file:
   a. Find the `dataSource` tag which is used for Codar database configuration.
   b. Add the following after the line that ends with `</security>`:

      Oracle:

      ```
      <validation>
      <check-valid-connection-sql>select 1 from DUAL</check-valid-connection-sql>
      <validate-on-match>false</validate-on-match>
      </validation>
      ```

      MS SQL or PostgreSQL:

      ```
      <validation>
      <check-valid-connection-sql>select 1</check-valid-connection-sql>
      <validate-on-match>false</validate-on-match>
      ```

```
        </validation>
```

3. Start the Codar service, see "Start Codar" on page 59.

Appendix A: Codar Console properties

This section lists and describes the properties that can be configured for the Codar Console, which are located in one of the following files:

- `CSA_HOME\jboss-as\standalone\deployments\csa.war\`
  `WEB-INF\classes\csa.properties`
- `CSA_HOME\jboss-as\standalone\deployments\csa.war\`
  `WEB-INF\web.xml`

The following areas contain properties that can be configured (for many properties, default values are provided):

- Authentication
- Account Lockout Mechanism
- Security banner attributes
- Notifications
- Approvers
- Security
- Codar keystore
- Service request processor scheduler
- Auditing
- Thread pool
- Process execution manager
- Lifecycle engine
- Approval engine scheduler
- LDAP cache scheduler
- Clustering
- Dynamic property
- Group approval
- Common access card
- Single sign-on
- Single Sign-On
- Process executor delegate
- Miscellaneous
- Operations Orchestration
- API authentication
- Topology designer
- Session timeout

After modifying the `csa.properties` file, restart Codar, see "Restart Codar" on page 60 .

Authentication

These properties are used for authentication. These properties are configured in `csa.properties`.

Property	Description
`csa.provider.hostname`	Required. The fully-qualified domain name of the system on which Codar is running. If you change this hostname, you must update the value of the `idm.codar.hostname` property in the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\spring\applicationContext.properties` file.
`csa.provider.port`	Required. The port used to connect to the system on which Codar is running. If you change this port, you must update the value of the `idm.codar.port` property in the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\spring\applicationContext.properties` file.
`csa.provider.rest.protocol`	Required. The protocol used by the REST API to connect to the system on which Codar is running. This attribute must be set to **https**. If you change this protocol, you must update the value of the `idm.codar.protocol` property in the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\spring\applicationContext.properties` file.
`csa.orgName.identifier`	Required. The provider organization identifier assigned to the organization who is providing this instance of the Codar Console. This attribute must be set to **`CSA-Provider.`**

Account Lockout Mechanism

These properties are configured in `csa.properties`.

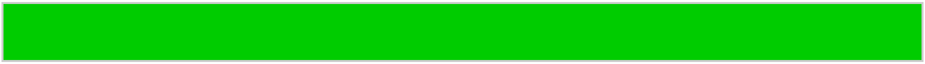
Property	Description
csa.login.lockout.enable	Required to enable the account lockout mechanism. To disable, set this property to false. It may be useful to disable account lockout in the case where an attacker continues to lock system accounts to cause denial of service, and the administrator is confident that all Codar Console users have very strong, secret passwords. **Default**: true
csa.login.maxFailedAttempts	The amount of failed login attempts that will lock the account **Note:** It is recommended that you set a lower amount of failed login

	attempts in clustered environments than in comparable non-clustered environments, since an attacker can distribute attacks over all nodes. **Default**: 3
csa.login.watchSeconds	The length of the watch period since the last failed login attempt after which the counter of failed login attempts will be reset. **Default**: 60 seconds
csa.login.lockSeconds	The length of the lockout period after which the account will be allowed to log in again. **Default**: 300 seconds (5 minutes)

Security banner attributes

The attributes in the following table are used by the Codar Console to enable or disable the display of a disclaimer upon logging in to the Codar Console and a color-coded banner that appears at the top and bottom of the Codar Console.

These properties are configured in `csa.properties`.

Attribute	Description
csa.provider.agency	By default, this attribute is commented out. When this attribute is commented out or does not contain a valid value, the login disclaimer and color-coded banners are not displayed for the Codar Console. If you want to enable the login disclaimer and color-coded banners, uncomment this attribute and set the value to **GOVERNMENT**. If set to any other value, the login disclaimer and color-coded banners are not displayed. To edit the disclaimer page, edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\static\template\disclaimerNote.jsp` file. To edit the disclaimer content, edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\msgs\messages_en.properties` file.
csa.provider. contentType	By default, this attribute is commented out. This attribute defines the color and content that displays in the security banner. The security banners appear at the top and bottom of the Codar Console. The following values are provided out-of-the-box: • UNCLASSIFIED. The banner is light green and contains no content. An example is shown below. 

Attribute	Description
	• UNCLASSIFIED_FOUO. For official use only. The banner is light green and displays the text "FOUO." An example is shown below. **FOUO** • UNCLASSIFIED_NOFORN. Not releasable to foreign nationals. The banner is light green and displays the text "NOFORN." An example is shown below. **NOFORN** • CONFIDENTIAL. The banner is light blue and displays the text "CONFIDENTIAL." An example is shown below. **CONFIDENTIAL** • CONFIDENTIAL_FOUO. The banner is light blue and displays the text "CONFIDENTIAL-FOUO." An example is shown below. **CONFIDENTIAL-FOUO** • CONFIDENTIAL_NOFORN. The banner is light blue and displays the text "CONFIDENTIAL-NOFORN." An example is shown below. **CONFIDENTIAL-NOFORN** • SECRET. The banner is red and displays the text "SECRET." An example is shown below. **SECRET** • TOPSECRET. The banner is orange and displays the text "TOPSECRET." An example is shown below. **TOPSECRET** To edit the banner content, edit the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\classes\msgs\messages_en.properties` file.

Notification

These property is used to enable or disable package promotion notification.

This property is configured in `csa.properties`.

Property	Description
`codar.PACKAGE_STATE_TRANSITION_NOTIFICATION`	Enables or disables package promotion notification. `true` enables package promotion notification. `text` disables package promotion notification. **Default**: `true`

Security

These properties are used to configure encrypted passwords (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses.

These properties are configured in `csa.properties`.

Property	Description
securityAdminPassword	Required. The encrypted password used by the out-of-the-box `admin` user (defined in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml` file). The admin user account is used for initial login to the Codar Console and can also be used to authenticate REST API calls. The password should be encrypted (see "Encrypt password" on page 60 for instructions on encrypting passwords). If you change this password, you must also update the password of any REST API calls that use this password. For more information about the REST APIs, see the *Codar API and CLI Reference Guide*.
securityCsaReporting UserPassword	Required. The encrypted password used by the out-of-the-box `csaReportingUser` user (defined in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml` file). The `csaReportingUser` user account is used when a subscription is ordered or modified and a field for the subscription includes a dynamically generated list. The dynamically generated list is a subscriber option property configured to use a dynamic query. The dynamic query uses this account to access Codar to determine the values that will appear in the list. This account has read-only access to Codar. The password should be encrypted (see "Encrypt password" on page 60 for instructions). If you change this password, you must also update the password of any REST API calls that use this password. For more information about the REST APIs, see the *Codar API and CLI Reference Guide*.
securityTransport UserName	Required. The out-of-the-box user used to authenticate REST API calls between the Marketplace Portal and Codar Console (it should not be used to log in to the Codar Console). If you change this username, you must update the value of the `idm.csa.username` property in the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\spring\applicationContext.properties` file. For more information about the integration user account, see

Property	Description
	"Change Codar out-of-the-box user accounts for Windows and Linux" on page 67. For more information about the REST APIs, see the *Codar API and CLI Reference Guide*.
securityTransportPassword	Required only if both the Cloud Service Automation and Codar licenses are used. The encrypted password used by the out-of-the-box `csaTransportUser` user (defined in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml` file). The `csaTransportUser` user account is used to authenticate REST API calls between the Marketplace Portal and Codar Console (it should not be used to log in to the Codar Console). The password should be encrypted (see "Encrypt password" on page 60 for instructions). If you change this password, you must update the value of the `idm.codar.password` property in the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\spring\applicationContext.properties` file. For more information about the integration user account, see "Change Codar out-of-the-box user accounts for Windows and Linux" on page 67. For more information about the REST APIs, see the *Codar API and CLI Reference Guide*.
securityOoInbound UserPassword	Required. The encrypted password used by the out-of-the-box `ooInboundUser` user (defined in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml` file). The ooInboundUser user account is used by Operations Orchestration to authenticate REST API calls with Codar (it should not be used to log in to the Codar Console). The password should be encrypted (see "Encrypt password" on page 60 for instructions). If you change this password, you must also update and use the same password for the `CSA_REST_CREDENTIALS` system account in Operations Orchestration (see "Operations Orchestration settings" on page 128 and the *Codar Installation and Configuration Guide*).
securityCdaInbound UserPassword	Required. The encrypted password used by the out-of-the-box `cdaInboundUser` user (defined in the `CSA_HOME\jboss-as\standalone\deployments\`

Property	Description
	csa.war\WEB-INF\applicationContext-security.xml file). The cdaInboundUser user account is used by Continuous Delivery Automation to authenticate REST API calls with Codar (it should not be used to log in to the Codar Console). The password should be encrypted (see "Encrypt password" on page 60 for instructions). If you change this password, you must also update and use the same password in Continuous Delivery Automation. For more information about this user account, see "Change Codar out-of-the-box user accounts for Windows and Linux" on page 67.
securityIdmTransport UserPassword	Required. The encrypted password used by the out-of-the-box idmTransportUser user (defined in the CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml file). The idmTransportUser user account is used to authenticate REST API calls (it should not be used to log in to the Codar Console). The password should be encrypted (see "Encrypt password" on page 60 for instructions). If you change this password, you must also update the following passwords (you must use the same password): • idmTransportUser property in the CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\classes\integrationusers.properties file. • Password of any REST API calls that use this password. For more information about this user account, see "Change Codar out-of-the-box user accounts for Windows and Linux" on page 67.
securityCatalog AggregationTransport UserPassword	Required. The encrypted password used by the out-of-the-box codarCatalogAggregationTransportUser user (defined in the CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml file). The codarCatalogAggregationTransportUser user account is used to authenticate catalog aggregation REST API calls with Codar (it should not be used to log in to the Codar Console). The password should be encrypted (see "Encrypt password" on page 60 for instructions).

Property	Description
	If you change this password, you must also update the password using the catalog aggregation registration REST APIs. For more information about this user account, see "Change Codar out-of-the-box user accounts for Windows and Linux" on page 67.
securityEncrypted SigningKey	Codar's encrypted signing key used to encrypt and decrypt authentication data passed between Codar and the Identity Management component. If you change this key, you must also update the `idm.encryptedSigningKey` property in the `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\spring\applicationContext.properties` file. The key should be encrypted (see "Encrypt password" on page 60for instructions. The encrypted key is preceded by `ENC` without any separating spaces and is enclosed in parentheses.
com.hp.ccue.consumption disallowedExtensions	A comma-delimited list of the file extensions that designate the types of documents or files that cannot be uploaded to the Codar Console. Default: exe,bat,com,cmd
csa.additionalSupported ExtensionsForImport	A comma-delimited list of the file extensions that designate the types of documents or files that can be uploaded to the Codar Console. The file extensions listed can be the sole extension of the file (for example, `mydocument.txt`, where `txt` is one of the listed file extensions) or the start of the file extension (for example, `mydocument.txt_3491767613`). Files can be uploaded using the Codar Console, the content archive tool, or the import API. See the *Codar Console Help* or *Codar API and CLI Reference Guide* for more information about using these features. The following extensions are automatically supported (and do not need to be defined by this property): jpg, jpeg, jpe, jfif, svg, tif, tiff, ras, cmx, ico, pnm, pbm, pgm, ppm, rgb, xbm, xpm, xwd, png, gif, bmp, cod, ief, json, xml, jsp, jspf. Default: (no default defined) Example: txt,log
csa.maxFileUploadSize	The maximum size of a file, in megabytes (MB), that can be uploaded to the Codar system using the Codar Console. If this property is not listed or is not set in the `csa.properties` file, the default maximum size of 50 MB is used.

Property	Description
	Default: 50 (MB)
csa.war.images.directory.byteLimit	A total size limit for all images or icons that are uploaded into `CSA_HOME/jbossas/standalone/deployments/csa.war/images`. The limit is used to prevent exhausting of server disk space through image upload in UI. Unit: bytes. Default: 500000000 bytes (500 MB)
csa.war.images.directory.smallFileByteOverhead	Used when computing space occupied by existing image/icon files (see above csa.war.images.directory.byteLimit). For each file in the images directory, a value of this property is added to its size to account for the overhead of small files on the file system. Unit: bytes. Default: 4096 bytes
enableSecurityWarning	Enables/disables the security warning messages for files that are uploaded or downloaded in the Cloud Service Management Console. Value is true or false. enableSecurityWarning is in the `CSA_HOME\jbossas\standalone\deployments\csa.war\offerings\config.json` file. Default: true

Codar keystore

These properties are used to configure information about Codar's keystore.

These properties are configured in `csa.properties`.

Property	Description
csaTruststore	Required. The Codar keystore that stores trusted Certificate Authority certificates. Default: No default specified **Example** `CSA_JRE_HOME/lib/security/cacerts` **Note:** Use only forward slashes (/) as your path separators.
csaTruststorePassword	Required. The encrypted password of the Codar keystore (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses. Default: No default specified **Example** `ENC(9eC7TTnB0uGOGK5U648UITcEV5AuV5T)`

Service request processor scheduler

These properties are used to configure the service request processor scheduler. The service request processor scheduler validates a consumer's requests, initiates the approval process, if configured, and maintains a request's status.

These properties are configured in `csa.properties`.

Property	Description
serviceRequestProcessorScheduler.maxInstancesToProcess	Optional. The maximum number of service requests the service request processor can process when it checks the start and end dates of submitted subscriptions. Default: 100
serviceRequestProcessorScheduler.period	Optional. How often, in milliseconds, the service request processor checks the start and end dates of submitted subscriptions. Default: 5000 (5 seconds)

Thread pool

These properties are used to configure thread pool.

Property	Description
com.hp.csa.service.process.ReleaseGateExecutor.DEPLOY_POOL_SIZE	Size of the thread pool for the release gate deploy action. Default: 2
com.hp.csa.service.process.ReleaseGateExecutor.CUSTOM_POOL_SIZE	Size of the thread pool for the release gate custom action. Default: 2
com.hp.csa.service.process.ReleaseGateExecutor.APPROVAL_POOL_SIZE	Size of the thread pool for the release gate approval action Default: 2
com.hp.csa.ReleaseGateExecutor.THREAD_WAKEUP_TIME	Thread pool wake up time for the release gate execution engine. The engine will sleep for the specified time in milliseconds Default:5000
com.hp.csa.ReleaseGateExecutor.THREAD_POOL_CORE_SIZE	Default thread pool size for all release gate actions Default: 2
com.hp.csa.service.process.ReleaseGateExecutor.REQUEST_MAX_SIZE	The maximum number of release gate instances that will be fetched by the engine at any point of time. Default: 30

Approvers

Property	Description
codar.ReleaseGate.Approver.MAX_LIMIT	The maximum number of approvers you can set to approve release gate action. Default: 10

Auditing

These properties are used to configure auditing.

These properties are configured in `csa.properties`.

Property	Description
csaAuditEnabled	Optional. Enable or disable auditing, which tracks user activities and system-generated events. Messages are logged to the `CSA_AUDIT_EVENT` table in the database. Default: true (enabled)
jboss.shutdown.	Required. This property is set during installation and *must not be changed*. The location

Property	Description
log.location	of the JBoss log file that records when the Codar service was stopped. Used for auditing purposes. Default: `CSA_HOME/jboss-as/bin/shutdown.log` **Note:** Use only forward slashes (/) as your path separators.

Process execution manager

These properties are used to configure the process execution manager. The process execution manager starts internal actions and Operations Orchestration flow actions, checks the status of process instances, and performs callback once the actions are completed.

These properties are configured in `csa.properties`.

Property	Description
`com.hp.csa.ProcessExecutor.THREAD_WAKEUP_TIME`	Optional. How often, in milliseconds, the process execution manager starts new process instances (which start Operations Orchestration flows) and checks the status of process instances. Default: 5000 (5 seconds)
`com.hp.csa.ProcessExecutor.THREAD_POOL_CORE_SIZE`	Optional. The maximum number of threads used to run process instances. Default: 2
`com.hp.csa.PEM.PARAM_PROCESS_INSTANCE_ID`	Optional. The token that stores the process instance ID and is used when Codar starts an Operations Orchestration flow. Default: `CSA_PROCESS_ID`
`com.hp.csa.PEM.PARAM_CONTEXT_ID`	Optional. The token that stores the artifact ID of the artifact that owns the action that executes the Operations Orchestration flow. Default: `CSA_CONTEXT_ID`

Lifecycle engine

These properties are used to configure the lifecycle engine. The lifecycle engine processes service instances and executes lifecycle actions.

These properties are configured in `csa.properties`.

Property	Description
`com.hp.csa.LifecycleExecutor.THREAD_WAKEUP_TIME`	Optional. How often, in milliseconds, the lifecycle engine checks for service components that it needs to transition.

Property	Description
	Default: 5000 (5 seconds)
`com.hp.csa.LifecycleExecutor.THREAD_POOL_SIZE`	Optional. The maximum number of threads used to transition service components. Default: 2
`application.lifecycle.stage.limit`	Optional. The maximum number of lifecycle stages that can be created. Default: 12 If you set it to a value greater than 12, it will default to a value of 12. That is, you cannot create more than 12 lifecycle stages.
`custom.roles.limit`	Optional. The maximum number of roles that can be created. Default: 100 If you set it to a value greater than 100, it will default to a value of 100. That is, you cannot create more than 100 roles.

Approval engine scheduler

This property is used to configure the approval engine scheduler. The approval engine scheduler checks each approver's response to a pending approval process to see if the process can be marked as completed and updates the decision and status of an approval process, as needed.

This property is configured in `csa.properties`.

Property	Description
`com.hp.csa.ApprovalDecisionMaker.THREAD_WAKEUP_TIME`	Optional. How often, in minutes, the approval engine scheduler checks for completion of an approval process to determine if an approval process should be approved or denied. Default: 1

LDAP cache scheduler

These properties are used to configure the LDAP cache scheduler. The LDAP cache scheduler checks the age of the user group cache and deletes it if it has expired.

For users who can log in to the Codar Console, certain actions require authorization (verification if the user belongs to a group). When authorization is requested for a user, Codar checks for group membership by using the cache. If the cache does not exist, LDAP is queried for the user's user groups which are temporarily cached to the database. After a configured expiration time, the cache is deleted. During a single session, the cache may be deleted and refreshed as needed.

These properties are configured in `csa.properties`.

Property	Description
`com.hp.csa.UserGroupExecutor.THREAD_WAKEUP_TIME`	Optional. How often, in minutes, the LDAP cache scheduler checks for user group caches that have expired. This number should be less than the value configured for `com.hp.csa.UserGroupExecutor.CACHE_EXPIRATION_TIME`. Default: 20
`com.hp.csa.UserGroupExecutor.CACHE_EXPIRATION_TIME`	Optional. How long, in minutes, LDAP user groups for a user are temporarily cached in the database before they are deleted. This time should be greater than the value configured for `com.hp.csa.UserGroupExecutor.THREAD_WAKEUP_TIME`. Default: 30
`com.hp.csa.UserGroupExecutor.UserGroupDeletionBatchSize`	Optional. The maximum number of user IDs that are deleted in a single batch from the cache. This number cannot be larger than 1,000. Default: 250

Clustering

This property is used to configure clustering.

This property is configured in `csa.properties`.

Property	Description
deploymentMode	Required. The mode in which Codar is running (single or clustered). When set to `single`, Codar runs in standalone mode (on a single instance) and all Codar services are run on this instance. When set to `clustered`, Codar runs in domain mode (in a clustered environment) and all Codar services are run on the master node. If you are using Microsoft SQL Server as your database, this property must be set to `single`. If you are running on Linux, this property must be set to `single`.

Property	Description
	Default: single

Dynamic property

These configuration properties are used to limit the amount of time to retrieve data and the amount of data retrieved when using a dynamic property. A dynamic property is a Dynamic Query value entry method for a subscriber option property that defines what information is retrieved. A dynamic property allows the Service Designer to list a dynamic set of values that change based on the user context (for example, the organization to which the user belongs).

These properties are configured in `csa.properties`.

Property	Description
`DynamicPropertyFetch.READ_TIMEOUT`	Optional. How long, in milliseconds, Codar attempts to fetch or retrieve data for dynamic properties. Default: 3000 (3 seconds)
`DynamicPropertyFetch.RESPONSE_SIZE`	Optional. The maximum amount of data, in bytes, that can be retrieved for dynamic properties. Default: 50000

Group approval

This configuration property is used when configuring a group approval template.

This property is configured in `csa.properties`.

Property	Description
`csa.group.numberOfApprovers`	Optional. The maximum number of members in an LDAP group used for approvals. For reasonable performance, do not specify more than ten (10) members. Default: 10

Common Access Card

This property is used to enable integration between Common Access Card and Codar.

This property is configured in `csa.properties`.

Property	Description
enableCAC	Optional. Enable integration between Common Access Card (CAC) and Codar, where the Common Access Card is used as an approval mechanism. To enable, this property must be uncommented and set to true. To disable, either comment out the property or set it to false. Default: (disabled)

Single sign-on

This property is used to enable integration between CA SiteMinder and Codar.

This property is configured in `csa.properties`.

Property	Description
enableSSO	Optional. Enable integration between CA SiteMinder and Codar, where the SiteMinder is used for single sign-on. To enable, this property must be uncommented and set to true. To disable, either comment out the property or set it to false. Default: (disabled)

Single Sign-On

This property is used to enable integration between Single Sign-On (Single Sign-On) and the Codar Console. Single Sign-On can be used when launching an application, such as the embedded Operations Orchestration, from the Codar Console. If you have installed or plan to integrate another single sign-on application or common access card with Codar, additional configuration to integrate with the Single Sign-On is required.

This property is configured in `csa.properties`.

Property	Description
enableHPSSO	Optional. Enable integration between Single Sign-On and the Codar Console. To enable, this property must be uncommented and set to true. To disable, either comment out the property or set it to false. This property is automatically set during installation.

Process executor delegate

These properties are used to configure the process executor delegate. The process executor delegate handles processing of the process instances. It discovers the ready instances, submits them to different thread pools for processing based on process definition and model type (sequenced or topology).

These properties are configured in `csa.properties`.

Property	Description
`com.hp.csa.service.process.ProcessExecutorDelegate.INTERNAL_POOL_SIZE`	Optional. The maximum number of threads used for processing internal executors (for example, clone patterns). Default: 2
`com.hp.csa.service.process.ProcessExecutorDelegate.EXTERNAL_POOL_SIZE`	Optional. The maximum number of threads used for processing external executors (for example, Operations Orchestration). Default: 2
`com.hp.csa.service.process.ProcessExecutorDelegate.CALLBACK_POOL_SIZE`	Optional. The maximum number of threads used by the callback pool. Default: 2
`com.hp.csa.service.process.ProcessExecutorDelegate.MONITOR_POOL_SIZE`	Optional. The maximum number of threads used by the monitor pool. Default: 2

Miscellaneous

The following is a miscellaneous property that does not fall under any specific category.

This property is configured in `csa.properties`.

Property	Description
com.hp.csa.aosMonitor. THREAD_WAKEUP_TIME	Optional. How often, in milliseconds, the background thread monitors plug-in processes. Default: 20000
com.hp.csa.TimeoutChecker. THREAD_WAKEUP_TIME	Optional. How often, in milliseconds, the background thread monitors for processes that have timed out. Default: 300000
com.hp.csa.ExportSvcOffering.THREAD_WAKEUP_TIME	Defines the background service wakeup time to export non-posted offerings, subscriptions and instances into Elasticsearch. When the CSA service starts, the background service wakes up. If there are no records to be exported to elasticsearch then the background services dies immediately. Otherwise the background service exports records into elasticsearch in the batches of the property defined in

Property	Description
	com.hp.csa.ExportSvcOffering.FETCH_SIZE. The background service continues to run until it processes all the non-posted records available in the CSA database. If the background service is not running, it wakes-up again according to the time defined in this property. The value of this property should be in milliseconds.
com.hp.csa.ExportSvcOffering.FETCH_SIZE	Defines the number of records to be processed at a time. The SQL used to fetch the records from the CSA database, uses this property value to limit the number of records that can be fetched from the database and then exported to Elasticsearch.
com.hp.csa.plugin.cloudos.util.TokenCache.TIMEOUT	Identity Management component token cache timeout, in milliseconds. Every REST call to CSA (such as for provisioning) is authenticated by Identity Management. CSA uses trustId to get the authentication token from Identity Management. Because these REST calls can be more frequent, this property allows you to define the cache timeout to prevent enormous sizes during the REST call’s authentication lifecycle. Default value: 300000 (5 minutes) Value 0 disables cache
com.hp.csa.import.BUILD_ARTIFACT_ RELATIONSHIP	Disables the artifact relationship section of the import/preview results.
loggerEnabled	Enables the logging filter for the legacy REST APIs, so that the requesting user and artifact information is logged.
csa.productPerspective	Determines which version of CSA has been installed: Enterprise or Codar.
jdbc.dialect	Holds explicitly set Hibernate dialect for a given database. Recommended values for the databases are: • MSSQL: org.hibernate.dialect.SQLServer2008Dialect • Oracle:

Property	Description
	org.hibernate.dialect.Oracle10gDialect - PostgreSQL: org.hibernate.dialect.PostgreSQLDialect

Operations Orchestration

These properties are configured in `csa.properties`.

The following properties configure the interaction between the Codar Console and Operations Orchestration. In the subscription event overview section of the **Operations** area in the Codar Console, selecting the Process ID opens Operations Orchestration to the detailed page of the selected process when these properties are configured.

Property	Description
OOS_URL	The URL used to access Operations Orchestration Central. This is the Operations Orchestration used for provisioning topology designs (Operations Orchestration version 10.21). Set this URL to the system on which Operations Orchestration version 10.21 is installed. For example, `https://<hostname>:8443`.
OOS_USERNAME	The username used to log in to Operations Orchestration Central. Set this username to admin.
OOS_PASSWORD	The encrypted password used by the user defined in `OOS_USERNAME` to log in to Operations Orchestration Central. Set this property to the encrypted value of the user defined in `OOS_USERNAME` (see "Encrypt password" on page 60). An encrypted password is preceded by `ENC` without any separating spaces and is enclosed in parentheses.
embedded.oo.root.dir	Location of the embedded Operations Orchestration when it is installed with CSA. This property is generated when embedded Operations Orchestration is installed during the CSA installation. This property is the only indicator of embedded Operations Orchestration, which is important mainly for uninstallation and upgrades. This property cannot be edited.

The following properties configure background services to monitor Operations Orchestration.

Property	Description
`com.hp.csa.oo.OOClient.SOCKET_TIMEOUT`	Optional. How long, in milliseconds, Codar keeps a socket open for SOAP-based communication with Operations Orchestration. Default: 60000
`com.hp.csa.OosMonitor.THREAD_WAKEUP_TIME`	Optional. How often, in milliseconds, the background thread monitors Operations Orchestration processes.

Property	Description
	Default: 60000
`com.hp.csa.service.process.OosMonitorDelegate.MONITOR_POOL_SIZE`	Optional. The maximum number of threads used by the monitor pool. Default: 2

Codar API authentication

These properties are used to configure authentication for the Codar 1.60 API. For details, see the *Codar API and CLI Reference Guide*.

Topology designer

These properties are used to configure the features of topology designs. Topology designs are built using components supported by various resource provider types and each component is bound to a specific provider type.

These properties are configured in `csa.properties`.

Property	Description
TopologyDesignProvisioning. TIMEOUT	Optional. The amount of time, in seconds, Codar attempts to provision or de-provision a topology design that is not based on an Helion OpenStack® provider (topology design provisioning and de-provisioning is orchestrated by interacting with resource providers corresponding to the components used in the design). If the time is exceeded, in the Operations area of the Codar Console, the subscription (to a service offering that is created from a topology design that is not based on an HPE Helion OpenStack® provider) will show a Subscription Status of `Failed` and a Service Instance Status of `Failed`. If you select the Events tab of the subscription, the event will show a Status of `Timeout`. If you select the Topology tab of the subscription, the topology view will show the status of the components in the service instance as their respective status just before the timeout occurred. HPE recommends that this value is set to the same value as the Operations Orchestration flow timeout value. Default: 7200 (2 hours)
OrchestratedTopologyDesignProvisioning. ProviderSelection.Enabled	Optional. Enable or disable the resource provider selection option (displaying or not displaying this option to a subscriber) for topology designs that are not based on an Helion OpenStack® provider. Default: true (enabled)
`csa.topology.expressDesignEnabled`	Optional. Enable or disable express designs in the topology designer. Express designs simplify the process of creating basic Helion OpenStack® topology designs. Default: false
`csa.topology.calloutsEnabled`	Optional. Enable or disable the `Pre-create Callout` and `Post-create Callout` properties of the Server Group Type component in the topology designer. See the *Codar Console Help* for more information about these properties. Default: false
`csa.topology.CloudOsSpecEnabled`	Optional. Enable or disable the **OpenStack** tab in the Create new design dialog in the topology designer. The tab allows the designer to select an OpenStack provider when creating a

Property	Description
	topology design. Default: false

Session timeout

This property is used to configure the Codar Console session.

This property is configured in `web.xml`.

Property	Description
session-timeout	Optional. The amount of inactivity, in minutes, that causes the Codar Console session to time out. Default: 60

Restart Codar service

After modifying the `csa.properties` file, restart Codar, see "Restart Codar" on page 60 .

Appendix B: Operations Orchestration settings

This section is provided as a reference only. The listed Operations Orchestration settings are configured in Operations Orchestration Studio and are used to integrate Operations Orchestration and Codar. These settings should have been configured as part of installing Codar. Information on how to configure these settings can be found in the *Codar Installation and Configuration Guide*.

The following areas contain settings that can be configured from Operations Orchestration Studio:

- "Remote action services" below
- "System accounts" below
- "System properties" on the next page

Remote action services

Setting	Description
`RAS_Operator_Path`	Required. The name and URL that accesses the RAS used by Operations Orchestration Central. HPE recommends the following value: `https://<FQDN>:9004/RAS/services/RCAgentService` where <*FQDN*> is the fully qualified domain name or IP address of the Operations Orchestration host. Do not use localhost in the URL. Using localhost does not work correctly even though it appears to work when you run Operations Orchestration Studio on the same machine as the RAS. RAS must be run on the same system as Operations Orchestration Studio. Running Operations Orchestration Studio on another machine produces errors and turns flows red with a cryptic error message about result assignments to result variables that do not exist.

System accounts

Setting	Description
`CSA_REST_CREDENTIALS`	Required. Credentials for Codar REST authentication. HPE recommends the Credentials are set to the following values: • **User Name**: ooInboundUser • **Password**: cloud **Note:** The **User Name** configured for the `CSA_REST_CREDENTIALS` System Account setting must match the **Override Value** (Operations Orchestration version 10.21) configured for the `CSA_OO_USER` System Property setting.

System properties

Setting	Description
`CSA_DMA_WorkflowTimeout`	Required. The amount of time, in seconds, to wait for a DMA workflow to complete. Default Property Value: 3600
`CSA_NA_CreateVlanScript`	Required. The name of the HPE Network Automation command script to create a VLAN that was imported when you integrated Network Automation with Codar. Default Property Value: HPN Create Vlan
`CSA_NA_DeleteVlanScript`	Required. The name of the Network Automation command script to delete a VLAN that was imported when you integrated Network Automation with Codar. Default Property Value: HPN Delete Vlan
`CSA_OO_USER`	Required. The user that communicates with Codar using the REST API. Default Property Value: ooInboundUser **Note:** The **Override Value** (Operations Orchestration version 10.21) configured for the `CSA_OO_USER` System Property setting must match the **User Name** configured for the `CSA_REST_CREDENTIALS` System Account setting.
`CSA_REST_URI`	Required. The URI used to communicate with Codar using the REST API. HPE recommends the following Property Value: `https://<codar_hostname>:8444/csa/rest`
`CSA_SiteScope_MonitoringLockId`	Required. SiteScope monitoring lock ID. Default Property Value: SiteScope Lock for Deploying Monitors
`CSA_SiteScope_RootMonitorGroup`	Required. The default name of the SiteScope root monitor group path. Default Property Value: Codar Monitors
`CSA_SiteScope_`	Required. The amount of time, in seconds, to wait before

System properties, continued

Setting	Description
`MonitoringSleepTime`	acquiring the SiteScope monitoring lock. This time may be increased if there are a large number of subscription requests. Default Property Value: 30
`CSA_ vCenterPropertyCollectionTimeout`	Required. How often, in seconds, properties are collected about a deployed virtual machine. Default Property Value: 1800

Appendix C: Identity Management configuration

If you are using the Identity Management component, the identity service and its components require configuration. Because it is a Spring Framework application, most of its configuration is defined in the `applicationContext.xml` file, although key attributes are externalized to the `applicationContext.properties` file. Both files are in `CSA_HOME\jboss-as\standalone\deployments\idm-service.war\WEB-INF\spring\`.

You should make most common configuration changes to the `applicationContext.properties` file. To avoid service disruptions, only advanced users who understand the Spring Framework should change the `applicationContext.xml` file.

You must also configure the Java Relying Party Library.

Note: You should always make a copy of a configuration file before editing it.

The following sections describe configuring the identity service and its components:

"External configuration" below

"Configure seeded authentication" on the next page

"Configure blacklist" on page 133

"Configure Java Relying Party Library" on page 133

"Internal configuration" on page 134

External configuration

Selected settings are pulled from the `applicationContext.properties` file, which you can override by an external properties file set as a JVM argument: `-Didm.properties="<`*`external_properties_filename`*`>"`. You can add this JVM argument to the `JAVA_OPTS` environment variable. Or you can edit the `standaloneconf.bat` file on Windows or `standalone.conf` file on Linux in `CSA_HOME\jboss-as\bin\` to add the JVM argument to `JAVA_OPTS` for the Codar JBoss container.

The table below describes the properties that are set in the properties file. These properties are required (although if you set the `idm.keystone.enabled` property to `false`, all other `idm.keystone*` properties in this table are ignored).

If you are integrating with Keystone, the `idm.keystone*` properties must match the Keystone network location, transport user credentials, and so on. All `idm.csa*` properties and all `ConvergedLdapAuthConfig` properties (which are listed in "ConvergedLdapAuthConfig" on page 136) must match the Codar network location and transport user credentials.

Property Name	Description
`idm.ssl.requireValidCertificate`	Flag indicating whether valid certificates are required: `true` or `false`
`idm.csa.protocol`	The protocol used to access the Codar instance: `http` or `https`
`idm.csa.hostname`	The hostname or IP address of the Codar server

Property Name	Description
`idm.csa.port`	The port number used by the Codar server
`idm.csa.username`	The user name for the Codar integration account
`idm.csa.password`	The password for the Codar integration account. For improved security, this value should be encrypted.
`idm.encryptedSigningKey`	The shared signing key for all token factory objects. For improved security, this value should be encrypted.
`idm.keystone.enabled`	Flag indicating whether secondary authentication through Keystone is enabled: `true` or `false`
`idm.keystone.required`	Flag indicating whether successful secondary authentication through Keystone is required for authentication to succeed: `true` or `false`
`idm.keystone.protocol`	The protocol used to access the Keystone instance: `http` or `https`
`idm.keystone.hostname`	The hostname or IP address of the Keystone server
`idm.keystone.port`	The port number used by the Keystone server. Typically 5000.
`idm.keystone.servicePath`	The service path where the Keystone service listens. The typical value is `v3`.
`idm.keystone.domainName`	The OpenStack domain name to use for all authentication on the Keystone server. The typical value is `Default`.
`idm.keystone.transportUsername`	The user name for the integration account used to communicate with Keystone and perform Helion OpenStack® or OpenStack operations.
`idm.keystone.transportPassword`	The password for the integration account used to communicate with Keystone and perform Helion OpenStack® or OpenStack operations. For improved security, this value should be encrypted.
`idm.keystone.transportProject`	The Keystone project name for the integration account. All Keystone users must belong to a project whose name exactly matches the Codar organization ID used to log in — including case (for example, a Keystone project name of `project_name` will not match an Codar organization ID of `PROJECT_NAME`.

Configure seeded authentication

The top-level configuration file for seeded authentication is specified by the `configFile` property of the `SeededAuthenticationProvider` bean defined in the `applicationContext.xml` configuration file. In the default configuration, this file is `seededorgs.properties`, but it can be changed. Each line in this file contains a key-value pair. The key is an Codar organization ID, and the value is the name of another properties file that contains the users for that organization. By default, the following organizations are configured to use the specified files.

Organization	User File
CSA_CONSUMER	csa-consumer-users.properties

You can define additional organizations or change the user file associated with any organization. Each line in each user file contains a key-value pair. The key is the user name, and the value is a comma-separated list of the password, granted authorities, and an optional flag indicating whether the account is enabled. For improved security, the *entire* value should be encrypted. Following is an example of a line from a user file that defines a user named `consumer` with the password `cloud` and granted the `SERVICE_CONSUMER` and `ROLE_REST` authorities.

```
consumer=cloud,SERVICE_CONSUMER,ROLE_REST,enabled
```

Configure blacklist

The blacklist contains users whom the identity service should never attempt to authenticate. In general, these are the Codar transport users and seeded Codar provider organization users, but you can edit this list. In the file, the blacklisted user name is associated with a Boolean value that indicates whether the user name is actually on the blacklist. A user might be temporarily removed from the blacklist by setting the Boolean value to `false`, but the value should generally be `true`. Following is the general format of each line in the file.

```
<username>= true
```

In the default configuration, the file contains the following contents.

```
admin = true
csaTransportUser = true
ooInboundUser = true
csaReportingUser = true
cdaInboundUser = true
csaCatalogAggregationTransportUser = true
```

This file should be updated to reflect any changes to the set of Codar transport users or seeded Codar provider organization users.

Configure Java Relying Party Library

The Java Relying Party Library is a set of classes provided by the identity service that abstract and simplify invoking the service from Java applications, such as Codar. You modify the properties listed in this section in the `CSA_HOME\jboss-as\standalone\deployments\csa.war\WEB-INF\applicationContext-security.xml` file. The `tokenFactory` property value should be the same for all `AuthenticationProvider` beans (listed in "Internal configuration" on the next page) in the identity service and in the Java Relying Party library.

IdentityServiceConfig

Configures the connection to the identity service.

Class: `com.hp.ccue.identity.rp.IdentityServiceConfig`

Property Name	Description
`protocol`	The protocol (`http` or `https`) to use to connect to the identity service
`hostname`	The hostname or IP address of the server running the identity service
`port`	The port number where the identity service is running, typically `8444`
`servicePath`	The path on the server to the identity service, typically `idm-service`

IdentityAuthenticationProvider

Abstracts the invocation of the identity service to perform authentication.

Class: `com.hp.ccue.identity.rp.IdentityAuthenticationProvider`

Property Name	Description
`templateFactory`	Creates the `RestTemplate` object that facilitates performing REST calls
`configuration`	Network configuration of the identity service to connect to perform authentication: an `IdentityServiceConfig` object
`tokenFactory`	The token factory to validate returned tokens
`tenantHeaderName`	The name of the HTTP header where the tenant name is passed. The default is `HP-Tenant-Name`

HeaderAuthenticationProvider

Performs authentication based on a token passed in an HTTP header.

Class: `com.hp.ccue.identity.rp.HeaderAuthenticationProvider`

Property Name	Description
`headerName`	The name of the HTTP header where the token is transferred
`tokenValidator`	The `TokenValidator` object to use to validate tokens

Internal configuration

The `applicationContext.xml` file defines the configuration of the classes in the identity service. The `tokenFactory` property value should be the same for all `AuthenticationProvider` beans (listed in the sections below) in the identity service and in the Java Relying Party library.

Note: Modify this file only if you cannot express the necessary configuration change in the `applicationContext.properties` file. The `applicationContext.xml` file must follow the syntax rules specified by the Spring Framework. In the following tables, the default values are used if no values are provided in the configuration file. You can configure items marked as externalized in the `applicationContext.properties` file.

InfinispanTokenStore

Defines the persistence mechanism for request tokens. Most attributes of this object define how the identity service behaves in high availability (HA) or clustered deployments.

Class: `com.hp.ccue.identity.ha.InfinispanTokenStore`

Property Name	Description

`lifetimeSeconds`
`lifetimeMinutes`
`lifetimeHours`
	Required. Time (in seconds, minutes, or hours) that an entry is permitted to remain in the token store. These properties determine the amount of time that the login page is valid. The lifetime as installed is 60 minutes. More permissive organizations should use a larger value; more restrictive organizations should use a smaller value.
	Default value: (None)
	Externalized: No

`clusterEnabled`	Required in a clustered environment. A flag indicating whether clustering should be enabled: `true` or `false` Default value: `false` Externalized: No
`clusterConfigFile`	Required in a clustered environment. The file name of the `jgroups.xml` configuration file that defines the cluster. Setting this property forces the `clusterEnabled` property to `true`. Default value: (None) Externalized: No
`configFile`	Required in a clustered environment. The file name of the Infinispan XML configuration file. The settings in this configuration file override the values in the `clusterEnabled` and `clusterConfigFile` properties. Default value: (None) Externalized: No

JwtTokenFactory

Defines how tokens are created.

Class: `com.hp.ccue.identity.domain.JwtTokenFactory`

Property Name	Description
`lifetimeMinutes`	Required. The lifetime of the token, in minutes. The lifetime as installed is 30 minutes. Reducing this value will render tokens invalid faster and thus requires a more-frequent token refresh, which might reduce performance. Increasing this value allows tokens to last longer, which might allow someone who has intercepted a valid token to access the system for a period of time. Default value: (None)

Property Name	Description
	Externalized: No
defaultTypeName	Optional. Default type of JWT token to create: PLAINTEXT, SIGNED, or ENCRYPTED Default value: PLAINTEXT Externalized: No
signingKey	Required if defaultTypeName is set to SIGNED. This is a Base64-encoded byte array representing the key used to sign signed tokens. If defaultTypeName is set to SIGNED, this value must be the same for all components that validate tokens. For improved security, this item should be encrypted. Default value: (None) Externalized: idm.encryptedSigningKey
refreshEnabled	Optional. Boolean value indicating whether token refresh is enabled: true or false. The recommended value is true. Default value: true Externalized: No

ConvergedLdapAuthConfig

Defines the configuration for connecting to an Codar server to get LDAP configuration information. The idm.csa* external properties (which are listed in the *External Configuration* section above) and all ConvergedLdapAuthConfig properties must match the Codar network location and transport user credentials.

Class: com.hp.ccue.identity.ldap.ConvergedLdapAuthConfig

Property Name	Description
providerProtocol	Required if using ActiveDirectory or LDAP. http or https, depending on the protocol used by the Codar instance Default value: (None) Externalized: idm.csa.protocol
providerHostname	Required if using ActiveDirectory or LDAP. Hostname or IP address of the Codar server Default value: (None) Externalized: idm.csa.hostname
providerPort	Required if using ActiveDirectory or LDAP. Port number used by the Codar server Default value: (None) Externalized: idm.csa.port
securityTransportUsername	Required if using ActiveDirectory or LDAP. Username for the Codar

Property Name	Description
	integration account Default value: (None) Externalized: `idm.csa.username`
`securityTransportPassword`	Required if using ActiveDirectory or LDAP. Password for the Codar integration account Default value: (None) Externalized: `idm.csa.password`

ConvergedActiveDirectoryAuthenticationProvider and ConvergedLdapAuthenticationProvider

Performs authentication with Active Directory and LDAP authentication mechanisms.

Class: `com.hp.ccue.identity.ldap.ConvergedActiveDirectoryAuthenticationProvider`, `com.hp.ccue.identity.ldap.ConvergedLdapAuthenticationProvider`

Property Name	Description
`config`	Required if using ActiveDirectory or LDAP. The `ConvergedLdapAuthConfig` that represents the Codar server to use to get the LDAP configuration for each organization Default value: (None) Externalized: No
`tokenFactory`	Required if using ActiveDirectory or LDAP. The token factory for creating identity tokens in response to successful authentications Default value: (None) Externalized: No

SeededAuthenticationProvider

Performs seeded authentication.

Class: `com.hp.ccue.identity.seeded.SeededAuthenticationProvider`

Property Name	Description
`configFile`	Required if using seeded authentication. Typically `seededorgs.properties`, which is the file that defines the seeded organizations Default value: (None) Externalized: No

Property Name	Description
`tokenFactory`	Required if using seeded authentication. The token factory for creating identity tokens in response to successful authentications Default value: (None) Externalized: No

IdentityAuthenticationProvider

Performs integration account authentication.

Class: `com.hp.ccue.identity.seeded.IntegrationAuthenticationProvider`

Property Name	Description
`configFile`	Required. Typically `integrationusers.properties`, which is the file that defines the seeded organizations Default value: (None) Externalized: No
`tokenFactory`	Required. The token factory for creating identity tokens in response to successful authentications Default value: (None) Externalized: No

MultiTenantAuthenticationProvider

Connects to mechanism-specific authentication providers.

Class: `com.hp.ccue.identity.authn.MultiTenantAuthenticationProvider`

Property Name	Description
`providers`	Required. List of `AuthenticationProvider` objects that provide mechanism-specific authentication Default value: (None) Externalized: No
`secondaryEnabled`	Required if using Keystone. Flag that indicates whether the secondary authentication path (Keystone) is enabled Default value: `false` Externalized: `idm.keystone.enabled`
`secondaryProvider`	Required if using Keystone. Reference to Authentication provider bean to use for secondary authentication path. The Keystone authentication provider is the only

Property Name	Description
	one that supports this type of usage. Default value: (None) Externalized: No
`secondaryRequired`	Required if using Keystone. Flag that indicates whether secondary (Keystone) authentication must succeed in order for authentication to be considered a success. Default value: `false` Externalized: `idm.keystone.required`

IdentityServiceImpl

The identity service implementation object.

Class: `com.hp.ccue.identity.service.IdentityServiceImpl`

Property Name	Description
`provider`	Required. Reference to the `AuthenticationProvider` bean to use to perform authentication. This is the `MultiTenantAuthenticationProvider` Default value: (None) Externalized: No
`tokenFactory`	Required. The token factory for creating identity tokens in response to successful authentications Default value: (None) Externalized: No
`blacklist`	A map associating usernames to Boolean values indicating whether they are blacklisted Default value: (None) Externalized: No
`blacklistFile`	The file containing the blacklist Default value: `blacklist.properties` Externalized: No
`queryService`	Required. The persistence service that provides all persistence operations. Default value: (None) Externalized: No
`trustFactory`	Required. The `TrustFactory` for validating all `Trust` objects. Default value: (None) Externalized: No

IdentityController

The controller object that provides the REST API for the identity service.

Class: `com.hp.ccue.identity.service.IdentityController`

Property Name	Description
`identityService`	Required. The `IdentityService` object that implements the identity service. You must set the value of this to the `IdentityServiceImpl` instance. Default value: (None) Externalized: No

KeystoneAuthenticationProvider

Uses Keystone (if used) to perform authentication.

Class: `com.hp.ccue.identity.keystone.KeystoneAuthenticationProvider`

Property Name	Description
`templateFactory`	Required. Creates the `RestTemplate` object that facilitates performing REST calls Default value: (None) Externalized: No
`configuration`	Required. Network configuration of the Keystone service to connect to in order to perform authentication: a `KeystoneConfig` object Default value: (None) Externalized: No
`tokenFactory`	Required. The token factory to validate returned tokens Default value: (None) Externalized: No

KeystoneConfig

Identifies the Keystone endpoint for authentication.

Property Name	Description
`protocol`	Optional if the default value is not acceptable. The protocol to access Keystone Default value: `http` Externalized: `idm.keystone.protocol`
`hostname`	Required. Optional if the default value is not acceptable. The hostname or IP address of the Keystone server

Property Name	Description
	Default value: (None) Externalized: `idm.keystone.hostname`
`port`	Optional if the default value is not acceptable. The port number for Keystone on `hostname` Default value: `5000` Externalized: `idm.keystone.port`
`servicePath`	Optional if the default value is not acceptable. The service path to the Keystone API on the Keystone server Default value: `v3` Externalized: `idm.keystone.servicePath`
`domainName`	Optional if the default value is not acceptable. The Keystone domain name under which all operations are performed Default value: `Default` Externalized: `idm.keystone.domainName`
`transportUsername`	Required. The username for the Keystone transport user Default value: (None) Externalized: `idm.keystone.transportUsername`
`transportPassword`	Required. The password for the Keystone transport user Default value: (None) Externalized: `idm.keystone.transportPassword`
`transportProject`	Required. The project for the Keystone transport user Default value: (None) Externalized: `idm.keystone.transportProject`

KeystoneSecondaryAuthenticationProvider

Uses Keystone (if used) to perform authentication.

Class: `com.hp.ccue.identity.keystone.KeystoneSecondaryAuthenticationProvider`

Property Name	Description
`keystoneConfigurations`	Required. Associative array mapping configuration identifiers to `KeystoneConfig` objects defining network configurations to connect to one or more Keystone services. Default value: (None) Externalized: No

Property Name	Description
`configurationFile`	Required. Filename for properties file that contains Keystone configurations. Default value: (None) Externalized: No
`tokenFactory`	Required. The token factory to validate returned tokens. Default value: (None) Externalized: No
`templateFactory`	Required. Creates the `RestTemplate` object that facilitates performing REST calls. Default value: (None) Externalized: No

RestTemplateFactoryImpl

Configures how REST services are invoked.

Class: `com.hp.ccue.identity.rest.RestTemplateFactoryImpl`

Property Name	Description
`wrapEnabled`	A flag that indicates whether the template factory should wrap JSON output in its specified root value or assume that incoming JSON is wrapped in the root value. This setting depends on the REST service being invoked. For template factories used to invoke Codar REST APIs, it should be set to `false`; for template factories used to invoke Keystone REST APIs, it should be set to `true`. Default value: `true` Externalized: No
`requireValidCertificate`	A flag that indicates whether the template factory should perform certificate validation and hostname verification (`true`) or ignore them (`false`). If this value is set to `true`, then the corresponding server host names for all beans that use that template factory must be given in a way that matches the certificate for that server (a fully-qualified domain name is generally required). Default value: `true` Externalized: `idm.ssl.requireValidCertificate`

TrustFactory

Configures how the Identity Management component trusts are created and validated.

Class: `com.hp.ccue.identity.domain.impersonation.TrustFactory`

Property Name	Description
`lifetime`	Required. The lifetime of a trust. Default value: 90 (days) Externalized: No
`lifetimeMinutes`	Required. Alternate setter for trust lifetime, expressed in minutes (write only). Default value: (None) Externalized: No
`lifetimeHours`	Required. Alternate setter for trust lifetime, expressed in hours (write only). Default value: (None) Externalized: No
`lifetimeDays`	Required. Alternate setter for trust lifetime, expressed in days (write only). Default value: (None) Externalized: No

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Configuration Guide (Codar 1.60)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to clouddocs@hp.com.

We appreciate your feedback!