HP Operations Orchestration

For the Windows ®, Solaris ®, and Linux operating systems

Software Version: 9.05

Administrator Guide

Document Release Date: June 2012

Software Release Date: June 2012

Legal Notices

Warranty

The only warranties for HP 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. HP 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 HP 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

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:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login 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 HP Software Support Online web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP 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 HP 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 an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Where Administrative Tasks are Documented

Some administrative tasks are performed from within HP OO Central and some are performed outside of Central. For instance, you configure and enable external authentication providers on the **Administration** tab in Central, but making other configuration changes to Central can require work with files outside of the Central Web application.

You can complete many administrative tasks from within Central, on the **Administration** tab. For example, on the Central **Administration** tab, you can manage flow runs; enable and configure user authentication by AD, LDAP, and Kerberos; and enable ROI reporting. If you can complete an administrative task entirely from within Central, you will find the procedure for performing the task in Central Help (and its PDF equivalent, the *Operations Orchestration Central User Guide*).

Information on administrative tasks related to configuring and supporting a Central server cluster is in *Operations Orchestration High Availability Guide*.

This *Operations Orchestration Administrator Guide* provides administrative concepts and procedures for completing administrative tasks that you cannot complete solely from within Central. Such administrative tasks include the following:

- Configuring HP OO for extended functionality
- Changing the maximum size of the `wrapper.log` file
- Enabling single sign-on for flows started with `Rsflowinvoke.exe`
- Changing Studio configurations in the `Studio.properties` file
- Backing up HP OO

Overview

Administering Operations Orchestration (HP OO, or OO) includes:

- Managing security, described in Security: Users, Groups, Capabilities, and Permissions.
- Enabling OO to run flows against remote machines and integrate them with other applications. For more information, see Configuring HP OO for extended functionality.
- Changing configurations for Central, including:
  - Configuring log file settings
  - Enabling single sign-on for flows started with the Java Flow Invoke tool
  - Enabling and disabling run-scheduling concurrency for Scheduler
  - Sizing Central for your particular workload
- Changing configurations for Studio, including the Studio host server, communications port number, and protocol used.
- Backing up OO

For information on administering flow runs, see Help for Central.

For information on supporting a Central server failover cluster and load-balancing, see *Operations Orchestration High Availability Guide* .

Default Ports Used by HP OO Components

By default, HP OO components use the following ports:

- Central: 8443

  If Central servers are clustered, a different port may be used.

- RAS: 9004

Security: Users, Groups, Capabilities, and Permissions

Many of the HP OO security features take place in the background. From the point of view of the OO administrator, author, and user, OO security deals with:

- Security of communications between OO system components and between those components and the flows' target systems.

  The aspect of this that is relevant to authors is the use of the HTTPS protocol and SSH for OO communications.

- User authentication, or logging in.

  You can configure OO to use external Active Directory, LDAP, or Kerberos authentication of user logins. To accomplish this configuration, you use the Central **Administration** tab. For information on doing so, see Help for Central.

  Note: In order to make communications secure, you can configure Active Directory to run over the Secure Sockets Layer, using the LDAPS protocol. For information on doing so, see Configuring Active Directory or LDAP over SSL (LDAPS protocol).

- Managing OO users and groups and controlling the actions they can perform in Central and Studio.

  In OO, groups are the unit through which users' ability to perform actions and the rights to perform those actions on particular flows, operations, and system accounts are controlled:

  - OO *capabilities* are sets of actions in OO that users can perform on Library objects, such as *authoring* (creating, checking in and out, and modifying), *promoting* a set of flows from a staging installation of Central to a production installation of Central, or *scheduling* flows.

    To give your flow authors the capability to author flows, you might create a group "Authors," which you would assign the AUTHOR capability.

    Note: No one can change the capabilities of the ADMINISTRATOR, AUDITOR, EVERYBODY, or PROMOTER group. These groups have special functions, which need to be protected from configuration changes:

    - ADMINISTRATOR

      This group has all the capabilities that are available in OO.

    - AUDITOR

      This group has the read capability throughout OO.

    - EVERYBODY

This the baseline group to which every OO user belongs. It has no capabilities. Therefore, to define the capabilities for any OO user a capability, you must add him or her to another group whose capabilities are either given or can be changed.

- PROMOTER

  The PROMOTER group exists specifically to publish repository objects from one repository to another. The existence of this group enables control of the promotion of operations, flows, or configurational objects in the repository from the development environment to a testing or to a production environment.

- Access to flows and other Library objects, such as operations, system accounts, and domain terms. Access permissions that are granted are Read, Write, Execute, and Link To.

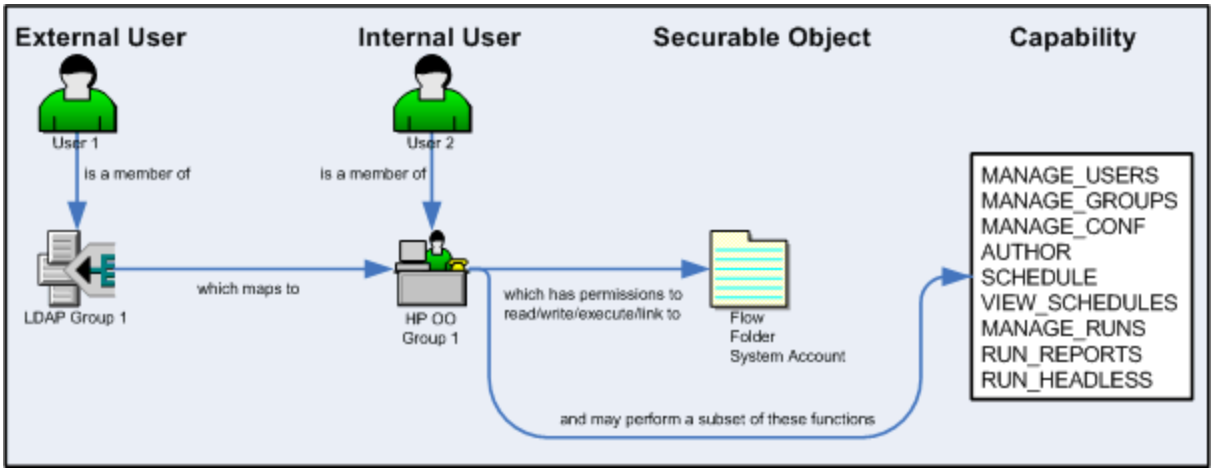
  For example:

  - For an author to make and test changes to a flow, the group that he or she is a member of must have the AUTHOR capability as well as the Read, Write, and Execute permissions for the flow. To add an operation (including a flow, which is a type of operation) to a flow, the author's group must have the Read and Link To permissions for the operation.
  - To run a flow (whether from within or outside the Central web application), a user must be a member a group that has Read and Execute permissions for the flow and all its operations and subflows.

Note: Access permissions are cumulative, so a user could have necessary permissions to run flow X if he or she had Read permission for flow X through membership in one group and Execute permission for flow X through membership in a different group.

Authors assign permissions for flows and associated objects in Studio. For information on doing so, see Help for Studio.

The following graphic shows how the concepts of users, groups, capabilities, and permissions interact to let administrators and authors define how individuals can react with which objects.



Components of access control in OO

- Protecting the confidentiality of sensitive credentials.

  System accounts are OO objects that make *impersonation* possible: a user can invoke credentials that are required for authentication when running a flow in a particular location,

without the user's being able to see the credentials that he or she is using to run the flow. As a result:

- Flow users can run flows wherever necessary without having to enter the credentials necessary for accessing the flows' targets
- The credentials remain protected.

Configuring Active Directory or LDAP over SSL (LDAPS protocol)

Computers ordinarily communicate with Active Directory or Lightweight Directory Access Protocol (LDAP, a clear-text protocol). To encrypt communications, you can set OO to communicate with Active Directory or LDAP using Secure Sockets Layer (SSL) communication. The LDAPS protocol is the LDAP protocol encrypted with SSL.

Configuring AD or LDAP to communicate with SSL requires the CA certificate that signed the AD/LDAP servers' SSL certificate. To configure AD or LDAP or Active Directory via SSL communication, you import the CA certificate into OO. The following procedure explains how to do so.

This procedure:

- Assumes that AD/LDAP over SSL has already been configured.
- Requires use of the Java keytool utility, which is included in the Java Runtime Environment (JRE). The JRE is installed with Central. If you are not familiar with the keytool utility, see the keytool documentation on the java.sun.com Web site.

To configure Central to communicate to LDAP or AD over SSL

1. If you are configuring Central to communicate with LDAP over SSL, export the LDAP server's CA certificate, then skip to step 14 ("Copy the exported certificate file to the OO Central server…") of this procedure. You might need to see your LDAP administrator for help with exporting a certificate.

   OR

   If you are configuring Central to communicate with AD over SSL, on one of the AD machines, start Microsoft Management Console (mmc.exe), and then complete the rest of this procedure to export the AD public certificate.

2. To add the Certificates snap-in:
   a. From the **File** menu, select **Add/Remove Snap-in**.
   b. In the **Add/Remove Snap-in** dialog, click **Add**.
   c. In the **Add Standalone Snap-in** dialog, select **Certificates** and click **Add**.
   d. Select **Computer Account** and then click **Next**.
   e. In the **Select Computer** dialog box, do one of the following:
      - If you are logged into the AD server, click **Finish**.
      - Otherwise, select **Another computer** and type in the name of a domain controller.

f. In the **Add standalone Snap-in** dialog, click **Close**, and in the **Add/Remove Snap-in** dialog click **OK**.

3. In the MMC console, open **Certificates (Local Computer)** and its subfolders **Personal\Certificates**.
4. In the right panel, find the certificate for the AD.

   The certificate should include:

   - The same name as the AD server.
   - An intended purpose of Client Authentication.
   - A Domain Controller certificate template.
5. Double-click the certificate to open it.
6. Click the **Certification Path** tab.
7. Click the ROOT certificate
8. Click **View Certificate**.

   A second **Certificate** dialog appears.
9. Click the **Details** tab.
10. Click the **Copy to File** button
11. When the Certificate Export Wizard starts, click **Next**.
12. In the **Export File Format** page, select **DER encoded binary X.509 (CER)** and click **Next**.
13. In the **File to Export** page, select the location and name of the exported certificate.

    The certificate file has a .cer extension.
14. Copy the exported certificate file to the OO Central server (the computer on which you have installed Central).
15. On the Central server, stop the RSCentral service.
16. In a command-line window, change directories to the OO home directory (%ICONCLUDE_HOME% on a Windows system or, on a Linux system, $ICONCLUDE_HOME), then to the jre1.6 directory's bin subdirectory.
17. To import the certificate, run the following command:

    ```
    keytool –importcert –keystore "%ICONCLUDE_HOME%\Central\conf\rc_
    keystore" –file <pathname_of_exported_cert> -alias <certificate_
    alias>
    ```

    where:

    - `<pathname_of_exported_cert>` is the pathname of the exported certificate.
    - `<certificate_alias>` is the certificate alias that you create using the –alias option.
18. When prompted for the certificate store's password, type **bran507025** (OO's default password for the rc_keystore keystore).
19. When you are prompted to confirm that this certificate should be trusted, type Yes.

20. To verify that the certificate was imported, use the following command:

    ```
    keytool –list –keystore "%ICONCLUDE_HOME%\Central\conf\rc_keystore"
    -alias <certificate_alias>
    ```

    where <certificate_alias> is the alias that you created when importing the certificate.

    You should see a summary of the certificate.

21. Restart the RSCentral service.

    You specify the URL for AD or LDAP in the OO Central web application, on the **System Configuration** subtab of the **Administration** tab. For information on how to do so, see Help for Central, in the topics on enabling AD or LDAP authentication of external users. The syntax for the URL is:

    ```
    LDAPS://<your_ldapsvr>:<port>
    ```

    where

    <your_ ldapsvr> is your LDAP or AD server

    <port> is the port that your LDAP or AD server uses for communication. The default port is 636.

    For example, if your AD is ad.mycompany.com and you have configured it to use the default port 636, the line should read as follows:

    ```
    LDAPS://ad.mycompany.com:636
    ```

Replacing a Security Certificate

The high-level procedure for replacing a security certificate breaks down into several sections:

- "Replacing the Default Security Certificate, and Modifying the RAS Keystore to Share Mutual SSL Authentication with Central" below
- "Replacing the Studio Certificate " on page 33
- "Testing the certificates" on page 34

Replacing the Default Security Certificate, and Modifying the RAS Keystore to Share Mutual SSL Authentication with Central

This procedure can be completed in four optional ways, depending on your needs:

- "Option 1a: Generate an SSL Certificate from a Trusted CA Server and Use an Existing Keystore " on next page
- "Option 1b: Generate an SSL Certificate from a Trusted CA Server and Create a New Keystore and Password" on page 21
- "Option 2a: Generate a Self-Signed SSL Certificate from Keytool.exe and Use an Existing Keystore" on page 27
- "Option 2b: Generate a Self-Signed SSL Certificate from Keytool.exe and Create a New Keystore and Password" on page 30

Note: In the following procedure, `%ICONCLUDE_HOME%\` represents the OO home directory. By default, `ICONCLUDE_HOME` is the environment variable representing the OO home directory.

The following procedure assumes that you have OpenSSL installed. For more information on OpenSSL, see the OpenSSL.org web site.

By default, OO has a self-signed certificate for each primary OO component, which resides at the following locations:

- Central: `%ICONCLUDE_HOME%\Central\conf\rc_keystore`
- Studio: `%ICONCLUDE_HOME%\Studio\conf\rc_keystore`
- RAS: `%ICONCLUDE_HOME%\RAS\java\default\webapp\conf\ras_keystore.jks`
- Load-Balancer: `%ICONCLUDE_HOME%\Clustering\apache\conf\ras_keystore`

- Scheduler: `%ICONCLUDE_HOME%\Scheduler\conf\rc_keystore` (for versions earlier than 9.03)

Option 1a: Generate an SSL Certificate from a Trusted CA Server and Use an Existing Keystore

There are two steps in this procedure:

- Replace the default security certificate for Central, Studio, and (for versions earlier than 9.03) Scheduler
- Modify the RAS keystore to share mutual SSL authentication with Central

Step 1: Replace the Default Security Certificate

You need to run this procedure separately for Central and Studio. For versions earlier than 9.03, you need to run it for Scheduler as well.

1. Make a backup copy of the following keystore:
   - `%ICONCLUDE_HOME%\Central\conf\rc_keystore`

   Note: The CN value must match the name of the Central application.

   Specifically for Central, you'll need a certificate that is purposed for both server authentication and client authentication. This is probably not the default SSL/Web Server certificate issued by the CA.

   When the CA Web site responds to your request, specify **Download CA certificate**.

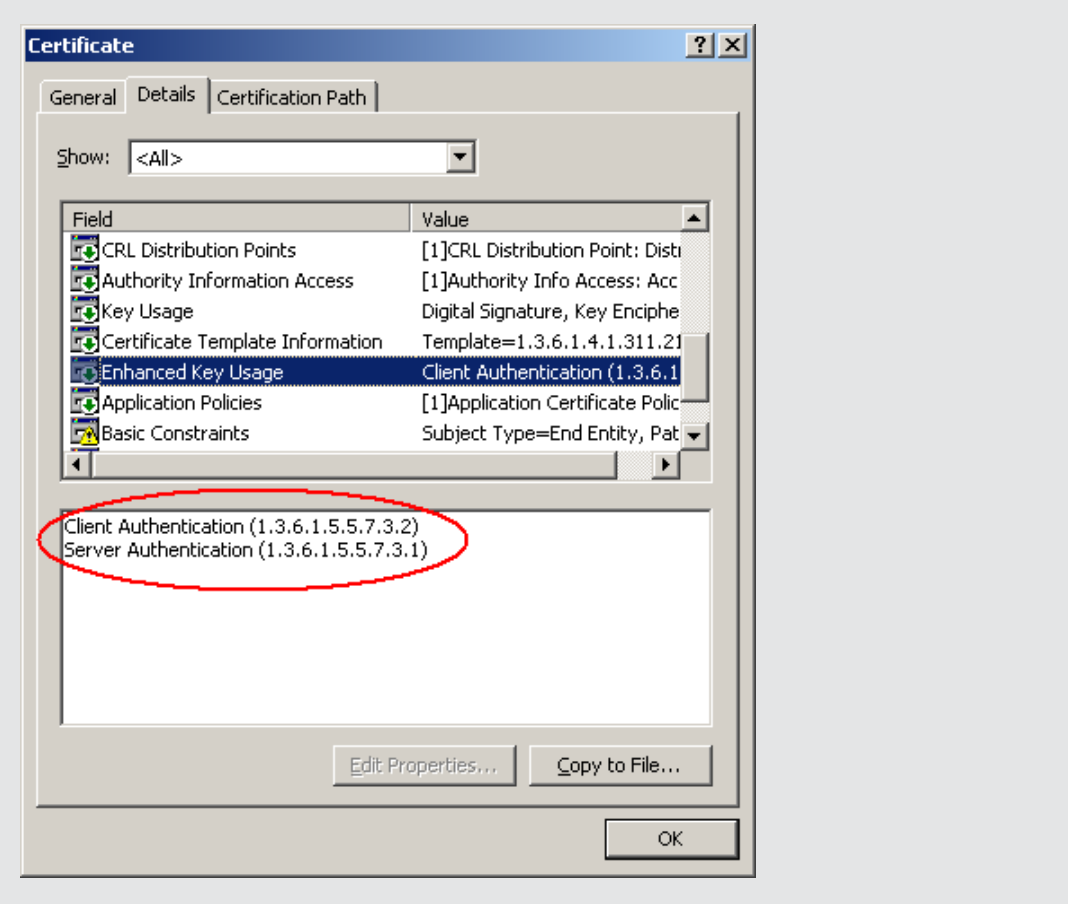
   Important! Import the certificates for each CA in the chain leading up to the CA that signed the certificate request. The CA certificates in the chain should be trusted certificates. If not, you will have to complete the additional Step 6.

   The certificate chain can be seen in the Certification Path. For example:

Note: The certificate must be issued for both server and client authentication. To verify this, complete the following steps:

a. Open `<certificate_name>.cer`.

b. On the **Details** tab, scroll down to the **Enhanced Key Usage** field.



2. If you already have a certificate with .pfx or .p12 extension, then go to the next step. If not, then you need to export the certificate with private key into PKCS12 format (.pfx,.p12). Also, make sure that you export the certificate with the alias name **pas**. For example, if the certificate format is PEM:

```
>openssl pkcs12 -export -in <cert.pem> -inkey <key.key> -out
pas.p12 -name pas
```

   If the certificate format is DER, add the `-inform DER` parameter after `pkcs12`.For example:

```
>openssl pkcs12 -inform DER -export -in <cert.pem> -inkey
<key.key> -out pas.p12 -name pas
```

   Make note of the password that you provide. You will need this password for the private key when you input the keystore passphrase later in this procedure.

3. Stop the RSCentral and RSJRAS services. If you are working with a version earlier than 9.03, stop the Scheduler service.

4. To import the certificate into a java rc_keystore, run the following command (you may have to change the path to specify the location where Central is installed for the rc_keystore:

```
>Java -cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
org.mortbay.jetty.security.PKCS12Import pas.p12 rc_keystore
```

- When asked for the input keystore passphrase, use the password that you gave when you exported the private key.
- For the output keystore passphrase, insert the `rc_keystore` password, which is "bran507025").

5. (Optional) To verify the keypair, run the following command, looking for the entry listed as `PrivateKeyEntry`:

   ```
   >keytool.exe -list -keystore rc_keystore -alias pas
   ```

   The output of the command should look something like the following:

   ```
   Keystore type: JKS

   Keystore provider: SUN

   Your keystore contains 2 entries

   le-f47ef679-334f-4116-b4ce-0bc288f76601, Dec 14, 2011,
   PrivateKeyEntry,

   Certificate fingerprint (MD5):
   92:B9:85:13:AD:A1:B2:2C:B2:CE:2B:7E:AD:B1:17:30

   testrootca, Dec 14, 2011, trustedCertEntry,

   Certificate fingerprint (MD5):
   2C:23:4F:44:76:5B:2A:35:71:60:B0:B7:0C:11:B0:F4
   ```

   Ensure that the listing shows PrivateKeyEntry.

6. (Optional) You need to complete this step only if the CA server from which you received the certificate is ***not*** a trusted server. Enter the command to trust all SSL connections based on CA:

   a. Import the CA certificate into the keystore, using the following command:

      ```
      >keytool -importcert -keystore rc_keystore -file
      <CACertificate.cer> -alias CACert
      ```

      where the `CACertificate.cer` file is the certificate of the untrusted CA authority from the certificate chain.

   b. At the prompt **Trust this certificate?** type **Yes**.

7. If you already changed the RASes' default certificates (using the procedure described in Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central), in order to import the new public keys from all of the RASes, you will need to do the following:
   a. Export the RAS's public key certificate, using the following command:

      ```
      >keytool -export -keystore ras_keystore.jks -alias ras -file
      RASPublicCert.cer
      ```

   b. You will need to delete the existing ras certificate from Central's keystore, using the following command:

      ```
      >keytool -delete -alias ras -keystore rc_keystore
      ```

c. After that, you can import the new RAS certificate with the same alias as it was before (ras), using the command:

```
>keytool –importcert –keystore rc_keystore –file
<RASPublicCert.cer> -alias ras
```

This step adds a new alias for RAS authentication using the same certificate.
If you haven't changed the RAS certificate yet, but you plan on changing it, then you need to go back to this step and perform this procedure ***after*** you successfully changed the RAS certificate, following the procedure described below, Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central.

8. Repeat the process so that it has been done for both Central and Studio, and for versions earlier than 9.03, also for Scheduler.

Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central

In this step, you will modify the RAS keystore to share mutual SSL authentication with Central, using the new Central certificate that you just created and also you will import a new RAS certificate.

1. To import the pkcs12 into a java keystore utilizing a jetty tool, run the following command (you may have to change the path to where RAS is installed):

   ```
   >Java –cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
   org.mortbay.jetty.security.PKCS12Import ras.p12 ras_keystore.jks
   ```

   When asked for input keystore passphrase, use the password that you provided when you exported the private key.

   For the output keystore passphrase, insert the `ras_keystore.jks` password, which is: "bran507025".

   > **Note:** If you want to use a different certificate for RAS, you can, but it is also enough to use a different alias for RAS and Central.

2. (Optional) To verify the keypair, run the following command, looking for the entry listed as `PrivateKeyEntry`.

   ```
   >keytool.exe -list -keystore ras_keystore.jks -alias ras
   ```

   The output of the command should look something like the following:

   ```
   ras, Mar 25, 2009, PrivateKeyEntry,

   Certificate fingerprint (MD5):
   3B:FC:EF:6F:53:1F:4D:D0:92:2C:FE:F2:63:15:73:D7
   ```

   Ensure that the listing shows `PrivateKeyEntry`.

3. If the CA server from which you received the certificate is not a trusted server, enter the command to trust all SSL connections based on CA:

a. Import the CA certificate into the keystore, using the following command:

```
>keytool -importcert -keystore rc_keystore -file
<CACertificate.cer> -alias CACert
```

where the `CACertificate.cer` file is the certificate of the untrusted CA authority from the certificate chain.

b. At the prompt **Trust this certificate?** type **Yes**.

4. To import all Central Servers Public Certificates:
   a. Export Central's public key certificate, using the following command:

```
>keytool -export -keystore rc_keystore -alias pas -file
CentralServerCertificate.cer.
```

   b. You will also need to delete the existing pas alias from `ras_keystore.jks` using the command:

```
>keytool -delete -alias pas -keystore ras_keystore.jks.
```

   c. Use the following command:

```
>keytool -importcert -keystore ras_keystore.jks -file
<CentralServerCertificate.cer> -alias <pas>.
```

   You might need to delete the existing certificate because the same alias already exists in the keystore. Use the following command:

```
>keytool -delete -alias pas -keystore ras_keystore.jks
```

   d. At the prompt **Trust this certificate?** type **Yes**.

5. Start the RSCentral and RSJRAS services.

Option 1b: Generate an SSL Certificate from a Trusted CA Server and Create a New Keystore and Password

There are two steps in this procedure:

- Replace the default security certificate for Central, Studio, and (for versions earlier than 9.03) Scheduler
- Modify the RAS keystore to share mutual SSL authentication with Central

Step 1: Replace the Default Security Certificate

You need to run this procedure separately for Central and Studio. For versions earlier than 9.03, you need to run it for Scheduler as well.

1. Make a backup copy of the following keystore:
   - `%ICONCLUDE_HOME%\Central\conf\rc_keystore`

Note: The CN value must match the name of the Central application.

Specifically for Central, you'll need a certificate that is purposed for both server authentication and client authentication. This is probably not the default SSL/Web Server certificate issued by the CA.

When the CA Web site responds to your request, specify **Download CA certificate**.

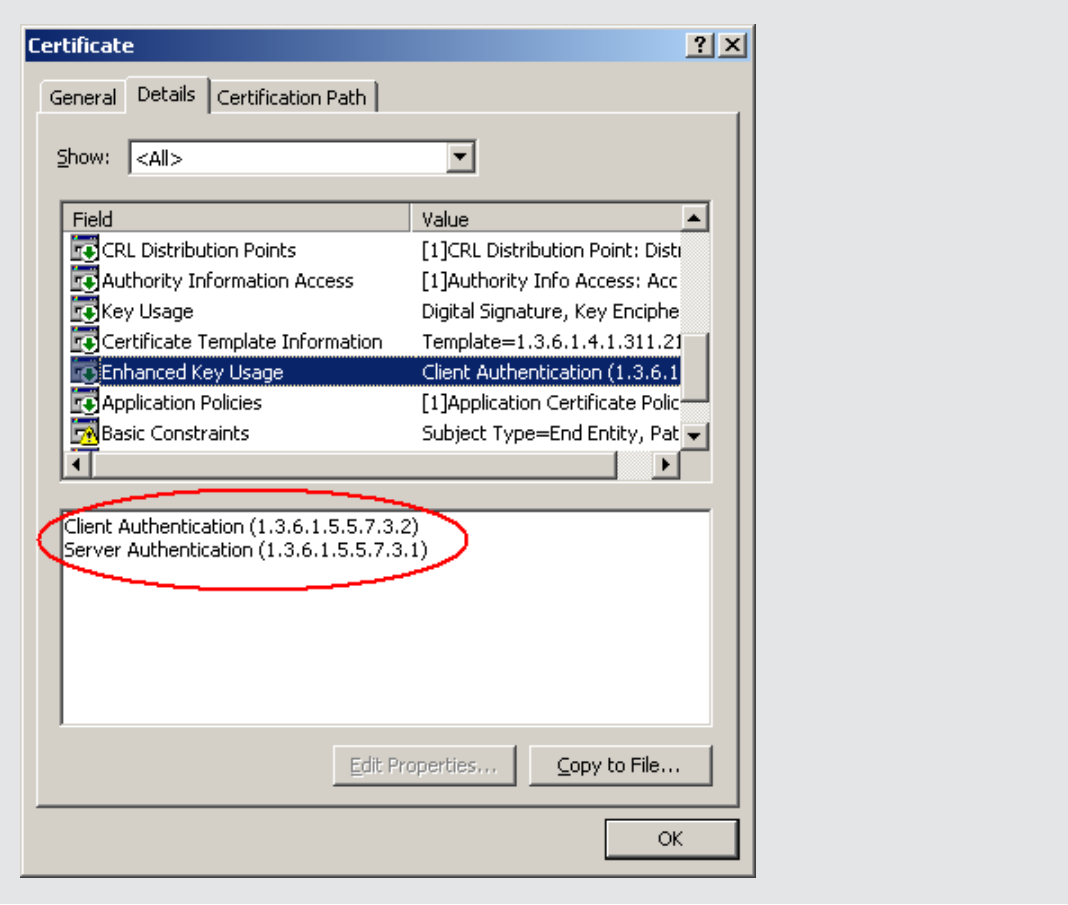
Important! Import the certificates for each CA in the chain leading up to the CA that signed the certificate request. The CA certificates in the chain should be trusted certificates. If not, you will have to complete the additional Step 6.

The certificate chain can be seen in the Certification Path. For example:

Note: The certificate must be issued for both server and client authentication. To verify this, complete the following steps:

a. Open `<certificate_name>.cer`.

b. On the **Details** tab, scroll down to the **Enhanced Key Usage** field.



2. If you already have a certificate with .pfx or .p12 extension, then go to the next step. If not, then you need to export the certificate with private key into PKCS12 format (.pfx,.p12). Also, make sure that you export the certificate with the alias name **pas**. For example, if the certificate format is PEM:

```
>openssl pkcs12 -export -in <cert.pem> -inkey <key.key> -out
pas.p12 -name pas
```

If the certificate format is DER, add the `-inform DER` parameter after `pkcs12`.For example:

```
>openssl pkcs12 -inform DER -export -in <cert.pem> -inkey
<key.key> -out pas.p12 -name pas
```

Make note of the password that you provide: You will need this password for the private key when you input the keystore passphrase later in this procedure.

3. Stop the RSCentral and RSJRAS services. If you are working with a version earlier than 9.03, stop the Scheduler service.
4. Create a new keystore with public and private certificates. A command like the following can be used:

```
>keytool -genkey -keyalg RSA -alias selfsigned -keystore
keystore.jks -storepass password -validity 360 -keysize 2048
```

5. To import the certificate into the new java keystore, run the following command (you may have to change the path to specify the location where Central is installed for the new keystore:

   ```
   >Java –cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
   org.mortbay.jetty.security.PKCS12Import pas.p12 <keystore_name>
   ```

6. To import the public keys from all of the RASes, use the following command:

   ```
   >keytool –importcert –keystore <keystore_name> –file
   <RASPublicCert.cer> -alias ras
   ```

   This step adds a new alias for RAS authentication using the same certificate.

7. Use the following command to obfuscate the keystore and private key passwords for use in the jetty.xml file:

   ```
   >Java –cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-
   6.1.14.jar";"%ICONCLUDE_HOME%\Jetty\lib\jetty-util-6.1.14.jar"
   org.mortbay.jetty.security.Password <password>
   ```

   Make note of the `OBF:<string>` value that this command shows; you will need it for editing the jetty.xml file. If the private key password and the keystore password are different, do so for both of them.

8. Edit jetty.xml as follows:

   - For your keystore password, update the `<Set name="Password">OBF:<string></Set>` with the `OBF:<string>` value you recorded in the preceding step.
   - For your private key password, update the `<Set name="KeyPassword">OBF:<string></Set>` with the `OBF:<string>` value from the preceding step.
   - Update `<Set name="Keystore"><SystemProperty name="jetty.home" default="." />/../Central/conf/rc_keystore</Set>` with the new file name and location.

9. Next, you update the passwords for your new keystore and certificate in the `central-secured.properties` file.

10. (Optional) If you replaced the default password, run the following command:

    ```
    >"%ICONCLUDE_HOME%\jre1.6\bin\java.exe" –cp "%ICONCLUDE_
    HOME%\Central\tools\lib\secure-props.jar";"%ICONCLUDE_
    HOME%\Central\WEB-INF\lib\dharma-commons.jar" ;" %ICONCLUDE_
    HOME%\Central\WEB-INF\lib\commons-cli-1.0.jar" ;"%ICONCLUDE_
    HOME%\Central\WEB-INF\lib\ant.jar" ;"%ICONCLUDE_HOME%\Central\WEB-
    INF\lib\log4j-1.2.8.jar" com.iconclude.dharma.tools.SecureProps –f
    "%ICONCLUDE_HOME%/Central/conf/central-secured.properties" –d
    dharma.security.ssl.trustStorePassword=<keystore password> -d
    dharma.security.ssl.keyStorePassword=<keystore_password>
    ```

- Depending on where Ant is installed, in the following step, you may need to change the command to reflect the actual location of the **ant.jar** archive in your installation. The **ant.jar** archive may not be located at **%ICONCLUDE_HOME%\Central\WEB-INF\lib**. In the 9.03-9.05 patch folder, there is an **ant.jar** that can be used.
- Update the password of the keystore only if you replaced the default with a new one in this procedure.

11. Repeat the process so that it has been done for both Central and Studio, and for versions earlier than 9.03, also for Scheduler.

Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central

In this step, you will modify the RAS keystore share mutual SSL authentication with Central, using the new Central certificate that you just created and a new RAS certificate.

1. To import the pkcs12 into a java keystore utilizing a jetty tool, run the following command (you may have to change the path to where RAS is installed):

   ```
   >Java -cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
   org.mortbay.jetty.security.PKCS12Import ras.p12 ras_keystore.jks
   ```

   When asked for input keystore passphrase, use the password that you provided when you exported the private key.

   For the output keystore passphrase, insert the keystore password (the default is "bran507025"). If you changed this password, you will need to complete the optional Steps 6 and 7 (editing the `jetty.xml` file and starting the RSCentral and RSJRAS services).

   Note: If you want to use a different certificate for RAS, you can, but it is also enough to use a different alias for RAS and Central.

2. (Optional) To verify the keypair, run the following command, looking for the entry listed as `PrivateKeyEntry`.

   ```
   >keytool.exe -list -keystore ras_keystore.jks -alias ras
   ```

   The output of the command should look something like the following:

   ```
   ras, Mar 25, 2009, PrivateKeyEntry,

   Certificate fingerprint (MD5):
   3B:FC:EF:6F:53:1F:4D:D0:92:2C:FE:F2:63:15:73:D7
   ```

   Ensure that the listing shows `PrivateKeyEntry`.

3. If the CA server from which you received the certificate is not a trusted server, enter the command to trust all SSL connections based on CA:

a. Import the CA certificate into the keystore, using the following command:

```
>keytool -importcert -keystore ras_keystore.jks –file
<CACertificate.cer> –alias CACert
```

where the `CACertificate.cer` file is the certificate of the untrusted CA authority from the certificate chain.

b. At the prompt **Trust this certificate?** type **Yes**.

4. To import all Central Servers Public Certificates:

a. Export Central's public key certificate, using the following command:

```
>keytool -export -keystore rc_keystore -alias pas -file
CentralServerCertificate.cer.
```

b. You will also need to delete the existing pas alias from `ras_keystore.jks` using the command:

```
>keytool -delete -alias pas -keystore ras_keystore.jks.
```

c. Use the following command:

```
>keytool –importcert –keystore ras_keystore.jks –file
<CentralServerCertificate.cer> -alias <CentralServerName>
```

d. At the prompt **Trust this certificate?** type **Yes**.

5. To obfuscate the keystore and private key passwords for use in the jetty.xml file, use the following command:

```
>Java –cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-
6.1.14.jar";"%ICONCLUDE_HOME%\Jetty\lib\jetty-util-6.1.14.jar"
org.mortbay.jetty.security.Password <password>
```

Make note of the `OBF:<string>` value, which you will need for editing the jetty.xml file. If the private key password and the keystore password are different, do this for both of them.

Next, you will edit the `jetty.xml` file.

> **Caution:** Update the password of the keystore (`"KeyPassword"`) only if you replaced the default keystore password in this procedure.

6. Edit `jetty.xml` as follows:

- Update the `<Set name="Password">OBF:<string></Set>` with the `OBF:<string>` value for your keystore password that you recorded.
- Update the `<Set name="KeyPassword">OBF:<string></Set>` with the `OBF:<string>` value for your private key password that you recorded.
- Update the `<Set name="Keystore"><SystemProperty name="jetty.home" default="." />/../Central/conf/rc_keystore</Set>` value with the new file name and location.

7. Start the RSCentral and RSJRAS services.

Option 2a: Generate a Self-Signed SSL Certificate from Keytool.exe and Use an Existing Keystore

There are two steps in this procedure:

- Replace the default security certificate for Central, Studio, and (for versions earlier than 9.03) Scheduler
- Modify the RAS keystore to share mutual SSL authentication with Central

Step 1: Replace the Default Security Certificate

You need to run this procedure separately for Central and Studio. For versions earlier than 9.03, you need to run it for Scheduler as well.

1. Make a backup copy of the following keystore:
   - `%ICONCLUDE_HOME%\Central\conf\rc_keystore`

   In this task, we will be using the standard java keytool to generate a self-signed certificate.
2. Stop the RSCentral and RSJRAS services. If you are working with a version earlier than 9.03, stop the Scheduler service.
3. Create a new keystore with public and private certificates. A command like the folowing can be used:

   ```
   >keytool -genkey -keyalg RSA -alias selfsigned -keystore
   keystore.jks -storepass password -validity 360 -keysize 2048.
   ```
4. To import the certificate into a java rc_keystore, run the following command (you may have to change the path to specify the location where Central is installed for the rc_keystore):

   ```
   >Java -cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
   org.mortbay.jetty.security.PKCS12Import pas.p12 rc_keystore
   ```
   - When asked for the input keystore passphrase, use the password that you gave when you exported the private key.
   - For the output keystore passphrase, insert the `rc_keystore` password, which is: “bran507025”.
5. To import the certificate into a java rc_keystore, run the following command (you may have to change the path to specify the location where Central is installed for the rc_keystore):

   ```
   >Java -cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
   org.mortbay.jetty.security.PKCS12Import pas.p12 rc_keystore
   ```
   - When asked for the input keystore passphrase, use the password that you gave when you exported the private key.
   - Be sure to set a password for the output keystore passphrase (by default, “bran507025”).
6. (Optional) To verify the keypair, run the following command, looking for the entry listed as PrivateKeyEntry:

```
>keytool.exe -list -keystore rc_keystore -alias pas
```

The output of the command should look something like the following:

```
Keystore type: JKS

Keystore provider: SUN

Your keystore contains 2 entries

le-f47ef679-334f-4116-b4ce-0bc288f76601, Dec 14, 2011,
PrivateKeyEntry,

Certificate fingerprint (MD5):
92:B9:85:13:AD:A1:B2:2C:B2:CE:2B:7E:AD:B1:17:30

testrootca, Dec 14, 2011, trustedCertEntry,

Certificate fingerprint (MD5):
2C:23:4F:44:76:5B:2A:35:71:60:B0:B7:0C:11:B0:F4
```

Ensure that the listing shows PrivateKeyEntry.

7. (Optional) You need to complete this step only if the CA server from which you received the certificate is ***not*** a trusted server. Enter the command to trust all SSL connections based on CA:
   a. Import the CA certificate into the keystore, using the following command:

      ```
      >keytool -importcert -keystore rc_keystore -file
      <CACertificate.cer> -alias CACert
      ```

   b. At the prompt **Trust this certificate?** type **Yes**.
8. If you already changed the RASes' default certificates (using the procedure described in Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central), in order to import the new public keys from all of the RASes, you will need to do the following:
   a. Export the RAS's public key certificate, using the following command:

      ```
      >keytool -export -keystore ras_keystore.jks -alias ras -file
      RASPublicCert.cer
      ```

   b. You will need to delete the existing ras certificate from Central's keystore, using the following command:

      ```
      >keytool -delete -alias ras -keystore rc_keystore
      ```

   c. After that, you can import the new RAS certificate with the same alias as it was before (ras), using the command:

      ```
      >keytool -importcert -keystore rc_keystore -file
      <RASPublicCert.cer> -alias ras
      ```

      This step adds a new alias for RAS authentication using the same certificate. If you haven't changed the RAS certificate yet, but you plan on changing it, then you need to get back to this step and perform this procedure ***after*** you successfully changed the RAS certificate following the procedure described below, Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central.
9. Repeat the process so that it has been done for both Central and Studio, and for versions earlier than 9.03, also for Scheduler.

Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central

In this step, you will modify the RAS keystore to share mutual SSL authentication with Central, using the new Central certificate that you just created and also you will import a new RAS certificate.

1. To import the pkcs12 into a java keystore utilizing a jetty tool, run the following command (you may have to change the path to where RAS is installed):

   ```
   >Java -cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
   org.mortbay.jetty.security.PKCS12Import ras.p12 ras_keystore.jks
   ```

   When asked for input keystore passphrase, use the password that you provided when you exported the private key.

   For the output keystore passphrase, insert the `ras_keystore.jks` password, which is: "bran507025".

   > **Note:** If you want to use a different certificate for RAS, you can, but it is also enough to use a different alias for RAS and Central.

2. (Optional) To verify the keypair, run the following command, looking for the entry listed as `PrivateKeyEntry`.

   ```
   >keytool.exe -list -keystore ras_keystore.jks -alias ras
   ```

   The output of the command should look something like the following:

   ```
   ras, Mar 25, 2009, PrivateKeyEntry,

   Certificate fingerprint (MD5):
   3B:FC:EF:6F:53:1F:4D:D0:92:2C:FE:F2:63:15:73:D7
   ```

   Ensure that the listing shows `PrivateKeyEntry`.

3. If the CA server from which you received the certificate is not a trusted server, enter the command to trust all SSL connections based on CA:

   a. Import the CA certificate into the keystore, using the following command:

      ```
      >keytool -importcert -keystore rc_keystore -file
      <CACertificate.cer> -alias CACert
      ```
      where the `CACertificate.cer` file is the certificate of the untrusted CA authority from the certificate chain.

   b. At the prompt **Trust this certificate?** type **Yes**.

4. To import all Central Servers Public Certificates:

   a. Export Central's public key certificate, using the following command:

      ```
      >keytool -export -keystore rc_keystore -alias pas -file
      CentralServerCertificate.cer.
      ```

   b. You will also need to delete the existing pas alias from

`>ras_keystore.jks` using the command: `keytool -delete -alias pas -keystore ras_keystore.jks`.

c. Use the following command:

```
>keytool –importcert –keystore ras_keystore.jks –file <CentralServerCertificate.cer> -alias <pas>.
```

You might need to delete the existing certificate because the same alias already exists in the keystore. Use the following command:

```
>keytool -delete -alias pas -keystore ras_keystore.jks
```

d. At the prompt **Trust this certificate?** type **Yes**.

5. Start the RSCentral and RSJRAS services.

Option 2b: Generate a Self-Signed SSL Certificate from Keytool.exe and Create a New Keystore and Password

There are two steps in this procedure:

- Replace the default security certificate for Central, Studio, and (for versions earlier than 9.03) Scheduler
- Modify the RAS keystore to share mutual SSL authentication with Central

Step 1: Replace the Default Security Certificate

You need to run this procedure separately for Central and Studio. For versions earlier than 9.03, you need to run it for Scheduler as well.

1. Make a backup copy of the following keystore:
   - `%ICONCLUDE_HOME%\Central\conf\rc_keystore`

   In this task, we will be using the standard java keytool to generate a self-signed certificate.
2. Stop the RSCentral and RSJRAS services. If you are working with a version earlier than 9.03, stop the Scheduler service.
3. Create a new keystore with public and private certificates. A command like the following can be used:

```
>keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass password -validity 360 -keysize 2048
```

4. To import the certificate into the new java keystore, run the following command (you may have to change the path to specify the location where Central is installed for the new keystore):

```
>Java –cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar" org.mortbay.jetty.security.PKCS12Import pas.p12 <keystore_name>
```

5. To import the public keys from all of the RASes, use the following command:

```
>keytool –importcert –keystore rc_keystore –file
<RASPublicCert.cer> -alias ras
```

This step adds a new alias for RAS authentication using the same certificate.

6. Use the following command to obfuscate the keystore and private key passwords for use in the jetty.xml file:

```
>Java –cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-
6.1.14.jar";"%ICONCLUDE_HOME%\Jetty\lib\jetty-util-6.1.14.jar"
org.mortbay.jetty.security.Password <password>
```

Make note of the `OBF:<string>` value that this command shows; you will need it for editing the `jetty.xml` file. If the private key password and the keystore password are different, do so for both of them.

7. Edit jetty.xml as follows:

   - For your keystore password, update the `<Set name="Password">OBF:<string></Set>` with the `OBF:<string>` value you recorded in the preceding step.
   - For your private key password, update the `<Set name="KeyPassword">OBF:<string></Set>` with the `OBF:<string>` value from the preceding step.
   - Update `<Set name="Keystore"><SystemProperty name="jetty.home" default="." />/../Central/conf/rc_keystore</Set>` with the new file name and location.

   Next, you update the passwords for your new keystore and certificate in the `central-secured.properties` file.

8. (Optional) If you replaced the default password, run the following command:

```
>"%ICONCLUDE_HOME%\jre1.6\bin\java.exe" –cp "%ICONCLUDE_
HOME%\Central\tools\lib\secure-props.jar";"%ICONCLUDE_
HOME%\Central\WEB-INF\lib\dharma-commons.jar" ;" %ICONCLUDE_
HOME%\Central\WEB-INF\lib\commons-cli-1.0.jar" ;"%ICONCLUDE_
HOME%\Central\WEB-INF\lib\ant.jar" ;"%ICONCLUDE_HOME%\Central\WEB-
INF\lib\log4j-1.2.8.jar" com.iconclude.dharma.tools.SecureProps –f
"%ICONCLUDE_HOME%/Central/conf/central-secured.properties" –d
dharma.security.ssl.trustStorePassword=<keystore password> -d
dharma.security.ssl.keyStorePassword=<keystore_password>
```

   - Depending on where Ant is installed, in the following step, you may need to change the command to reflect the actual location of the **ant.jar** archive in your installation. The **ant.jar** archive may not be located at **%ICONCLUDE_HOME%\Central\WEB-INF\lib**. In the 9.03-9.05 patch folder, there is an **ant.jar** that can be used.
   - Update the password of the keystore only if you replaced the default with a new one in this procedure.

9. Repeat the process so that it has been done for both Central and Studio, and for versions earlier than 9.03, also for Scheduler.

Step 2: Modify the RAS Keystore to Share Mutual SSL Authentication with Central

In this step, you will modify the RAS keystore share mutual SSL authentication with Central, using the new Central certificate that you just created and a new RAS certificate.

1. To import the pkcs12 into a java keystore utilizing a jetty tool, run the following command (you may have to change the path to where RAS is installed):

   ```
   >Java -cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-6.1.14.jar"
   org.mortbay.jetty.security.PKCS12Import ras.p12 ras_keystore.jks
   ```

   When asked for input keystore passphrase, use the password that you provided when you exported the private key.

   Make sure to set a password for output keystore passphrase (by default, "bran507025"). If you changed this password, you will need to complete the optional Steps 6 and 7 (editing the `jetty.xml` file and starting the RSCentral and RSJRAS services).

   Note: If you want to use a different certificate for RAS, you can, but it is also enough to use a different alias for RAS and Central.

2. (Optional) To verify the keypair, run the following command, looking for the entry listed as `PrivateKeyEntry`.

   ```
   >keytool.exe -list -keystore ras_keystore.jks -alias ras
   ```

   The output of the command should look something like the following:

   ```
   ras, Mar 25, 2009, PrivateKeyEntry,

   Certificate fingerprint (MD5):
   3B:FC:EF:6F:53:1F:4D:D0:92:2C:FE:F2:63:15:73:D7
   ```

   Ensure that the listing shows `PrivateKeyEntry`.

3. If the CA server from which you received the certificate is not a trusted server, enter the command to trust all SSL connections based on CA:

   a. Import the CA certificate into the keystore, using the following command:

      ```
      >keytool -importcert -keystore rc_keystore -file
      <CACertificate.cer> -alias CACert
      ```

   b. At the prompt **Trust this certificate?** type **Yes**.

4. To import all Central Servers Public Certificates:

   a. Export Central's public key certificate, using the following command:

      ```
      >keytool -export -keystore rc_keystore -alias pas -file
      CentralServerCertificate.cer.
      ```

   b. You will also need to delete the existing pas alias from ras_keystore.jks using the command:

      ```
      >keytool -delete -alias pas -keystore ras_keystore.jks.
      ```

c. Use the following command:

```
>keytool –importcert –keystore ras_keystore.jks –file
<CentralServerCertificate.cer> -alias <CentralServerName>
```

d. At the prompt **Trust this certificate?** type **Yes**.

5. To obfuscate the keystore and private key passwords for use in the jetty.xml file, use the following command:

```
>Java –cp "%ICONCLUDE_HOME%\Jetty\lib\jetty-
6.1.14.jar";"%ICONCLUDE_HOME%\Jetty\lib\jetty-util-6.1.14.jar"
org.mortbay.jetty.security.Password <password>
```

Make note of the `OBF:<string>` value, which you will need for editing the jetty.xml file. If the private key password and the keystore password are different, do this for both of them.

Next, you will edit the `jetty.xml` file.

> **Caution:** Update the password of the keystore (`"KeyPassword"`) only if you replaced the default keystore password in this procedure.

6. Edit `jetty.xml` as follows:
   - Update the `<Set name="Password">OBF:<string></Set>` with the `OBF:<string>` value for your keystore password that you recorded.
   - Update the `<Set name="KeyPassword">OBF:<string></Set>` with the `OBF:<string>` value for your private key password that you recorded.
   - Update the `<Set name="Keystore"><SystemProperty name="jetty.home" default="." />/../Central/conf/rc_keystore</Set>` value with the new file name and location.
7. Start the RSCentral and RSJRAS services.

Replacing the Studio Certificate

This step is optional. You need to replace the password of the keystore only if you replaced the default keystore password with a new one in Replacing the default security certificate.

Studio and Central each have their own copies of rc keystore, even when installed on the same machine. After making changes to the Central rc keystore, back up the Studio rc keystore, then write over the Studio rc keystore with the Central rc keystore.

By default, OO has a self-signed certificate for each primary OO component, which resides at the following locations:

- Central: `%ICONCLUDE_HOME%\Central\conf\rc_keystore`
- Studio: `%ICONCLUDE_HOME%\Studio\conf\ rc_keystore`
- RAS: `%ICONCLUDE_HOME%\RAS\java\default\webapp\conf\ras_keystore.jks`
- Load-Balancer: `%ICONCLUDE_HOME%\Clustering\apache\conf\ ras_keystore`

- Scheduler: `%ICONCLUDE_HOME%\Scheduler\conf\rc_keystore` (for versions earlier than 9.03)

To replace Studio's certificate

- Update Studio's secured.properties file with the following command:

```
>"%OO_HOME\jre1.6\bin\java.exe" -cp "%ICONCLUDE_
HOME%\Central\tools\lib\secure-
 props.jar";"%ICONCLUDE_HOME%\Central\WEB-INF\lib\dharma-
commons.jar"
 ;"%ICONCLUDE_HOME%\Central\WEB-INF\lib\commons-cli-1.0.jar"
 ;"%ICONCLUDE_HOME%\Central\WEB-INF\lib\ant.jar"
 ;"%ICONCLUDE_HOME%\Central\WEBINF\lib\log4j-
 1.2.8.jar"com.iconclude.dharma.tools.SecureProps -f
 "%ICONCLUDE_HOME%/Studio/conf/studio-secured.properties" -d
 dharma.security.ssl.trustStorePassword=<keystore password> -d
 dharma.security.ssl.keyStorePassword=<keystore_password>
```

Testing the certificates

To test the certificates

- In your web browser, open the Central web application with the usual URL:

`https://<host>:<ssl_port>/PAS`

where

`<host>` is the name of the Central server.

`<ssl_port>` is the port that the Central server uses for communication using SSL.

Configuring HP OO for Extended Functionality

Extended functionality in OO is the use of flows that can:

- Execute actions on machines that are on different domains, on the other side of firewalls, or even on different datacenters from the Central Web server (the machine on which you installed the Central Web application).
- Carry out actions that interact with an application whose application programming interface (API) cannot be accessed remotely and therefore requires a RAS on the application's server.

Such actions are carried out by Remote Action Service (RAS) operations. RAS operations are enabled, or hosted, by the RAS Web service, which can be installed (using the standalone RAS installer) on different subnets, servers, or datacenters from the Central server. By default, a RAS is installed during the Central Web application installation. A RAS operation requires a reference that directs it to the RAS installation that it needs to run.

If you install a RAS in addition to the default RAS, you necessarily install the second RAS standalone—that is, on a different machine from the Central server. In Studio, you will then configure a reference for the new, standalone RAS. The reference is made up of a name and the URL of the RAS. Any RAS operation that uses this standalone RAS must include a reference to the reference. (For information on adding a RAS reference in a RAS operation, see Help for Studio.)

The following considerations may affect how you install RAS:

- Where you need to install RAS and its content.

  The Central installation installs RAS on the Central server. However, to run an operation against a machine that is on a different domain or the other side of a firewall from the Web server, RAS must be installed on the machine against which you're going to run the operation.

  Note: When you install a standalone RAS in an AD domain, the RAS must run on an AD domain account in order to run .NET-base operations.

- Whether the application you will run the operation against is accessed through an API that cannot be accessed remotely (and that do not reside on the OO Central server).

  In the Studio Library, the **Description** tab of the **Properties** sheet of a folder in OO default content (flows and operations that come with OO) tells whether the flows and operations in the folder require a RAS. If they do, the folder gives this information under the heading "Deployment Requirements."

Therefore, after you install the Central Web server, you must also install a standalone RAS if you run operations that:

- A machine that is on a different domain or on the other side of a firewall from the Central Web server.

You only need to install RAS on one machine on the other domain or on the far side of the firewall in order to run a RAS-dependent operation on other machines there.

- Do interact with an application whose API cannot be accessed remotely.

For information on installing RAS and RAS content and testing the installations, see *Installing or Upgrading HP Operations Orchestration to 9.00 Windows and Linux Operating Systems* (or, on a Linux machine, the linux.README.txt file for RAS).

Changing Central Configurations

Some Central configurations you can change on the **Administration** tab of OO Central. Others, you change by making modifying various files.

Configurations that you can change on the Central **Administration** tab include:

- Which authentication providers are enabled and specific settings for how OO uses them.

  OO supports the following authentication providers:

  - Active Directory (AD)
  - Lightweight Directory Access Protocol (LDAP)
  - Kerberos

  For information on enabling authentication with one or more of these providers, see Help for Central. (Because topic names can change, search for “external authentication”.)

Central configurations that you change outside of Central include the following:

- Changing Anti-Samy input filter settings

  To prevent the introduction of malicious software, Central employs the Anti-Samy set of input filters. in the Central.properties file, you can specify the level of security provided by the Anti-Samy input filter system.

- The maximum size of the log files

  If you install Central as a Windows service, then by default the maximum size for Wrapper.log is 64 megabytes (MB). When the file reaches that size, the file begins to roll—that is, the oldest entry is deleted as each new entry is added. For information on changing the maximum size of Wrapper.log, see the procedure, To change the maximum size of the Jetty service Wrapper.log.

- Enabling flows started by RSFlowinvoke.exe to run without a new login

  Running without a new login being required is also known as single sign-on, or SSO.

- Enabling and disabling run-scheduling concurrency for Scheduler
- Sizing Central for your workload

  Sizing Central involves configuring:

  - The size of memory allocated to the Central process.
  - The maximum amount of RAM allowed for the Central server process.
  - The maximum number of threads to be allowed to parallel processing.

- Changing the timeout limit for RAS operations
- Changing whether the password will be erased when a repository containing system accounts is exported

Changing the Maximum Size of the Log Files

Central, the Central scheduling component (also known as "the Scheduler"), and the RAS service each has a log file, with the following names and locations within the OO home directory:

- Central

  `\Central\logs\Central_wrapper.log`

- Scheduler

  `\Central\logs\Scheduler\wrapper.log`

- RAS

  `\RAS\Java\Default\webapp\logs\wrapper.log`

You can change the maximum size that the log files can reach by setting the `wrapper.logfile.maxsize` property in each component's `wrapper.conf` file.

Note: In Linux installations, the file in Central is called `wrapper_linux.conf`, rather than `wrapper.conf`.

To change the maximum size of a log file

1. In the OO home directory, navigate to the appropriate one of the following locations (according to which component's `wrapper.log` file you want to configure) and then open `wrapper.conf`:
   - Central and Scheduler

     `\Central\conf\wrapper.conf`

     Note: In Linux installations, the file in Central is called `wrapper_linux.conf`, rather than `wrapper.conf`.

   - RAS

     `\RAS\Java\Default\webapp\conf\wrapper.conf`
2. Locate the property `wrapper.logfile.maxsize` and specify the maximum size in bytes that the log file should reach before it starts removing the oldest entries as it adds new ones (or rolling).

   You can abbreviate the size value of this property by adding k or m (for kilobytes or megabytes, respectively) to the end of the size.

   By default, the maximum size is 64 MB.

   Note: Setting the value to zero (0) disables rolling, and the file will grow indefinitely.
3. Save and close the file.

Enabling Single Sign-on for OO Central

Single sign-on (SSO) and lightweight single single sign-on (LWSSO) for OO Central make logging on to Central and starting flow runs easier:

- Once you have previously logged on to another HP product web client (such as the SM web client or the BSM web client) in a given web-browser session, you can open Central without having to log in again. SSO makes this possible by sharing a cookie between HP products web tiers that are accessed from a web browser.
- You can start a flow run from outside Central using the credentials of the account that is already logged on to the computer.

In addition, SSO and LWSSO can provide security and performance benefits.

LWSSO is somewhat simpler to enable and configure than SSO; for information on enabling LWSSO, see Enabling Lightweight SSO on OO Central.

Notes:

- SSO support in Central is based on Kerberos 5, and works with any compliant Kerberos 5 implementation. The following procedures on configuring machines for SSO assume that you are familiar with Kerberos fundamentals.
- Whether enabling SSO for starting flows from outside the Central Web application or for working within the Central Web application, you first configure the Central server to enable SSO. The tasks that you perform after that depend on which of those two capacities you are enabling SSO for. This section on enabling SSO is divided accordingly.
- SSO does not by itself enable users to do anything once signed in to Central. Any user must be a member of an OO internal group (or groups) that have the appropriate capabilities to perform any desired actions (and, in the case of running flows, access permissions for the flows). For instance, to view run-history reports, the user's group must have the RUN_REPORTS capability. And to start a flow run from outside Central, the user's group (or groups) must have both of the following:
  - The HEADLESS_FLOWS capability, which allows a user to start a flow from outside Central
  - The necessary access permissions (READ and EXECUTE) to that flow

    For information on setting access permissions for flows and other Central repository (Library) objects, see the *OO Studio Authoring Guide*.

The procedures for enabling SSO for the Central server vary according to which type of key distribution center (KDC) Central is to use: a Windows KDC (Active Directory, which supports the Kerberos 5 specification) or a Linux KDC:

- Active Directory, using the procedures in Enabling single sign-on for OO Central using Windows AD

- Kerberos (MIT KDC), using the procedures in Enabling single sign-on for OO Central using MIT KDC

Note: The preceding two sections include instructions that are specific to enabling SSO for access to Central via a network load balancer.

After enabling SSO for OO Central, you can do either or both of the following:

- Enable SSO for starting a flow from outside Central
- Enable SSO for the Central Web application

  You can enable SSO for the Central Web application for:

  - Internet Explorer or Firefox, on a machine running the Windows operating system.
  - Firefox, on a machine running Linux.

These discussions and procedures assume that you are familiar with Active Directory administration and configuration issues, and Kerberos concepts and tools.

Component	In AD examples	In Linux/Kerberos examples
OO Central server	mycentral.greendomain.ad	mycentral.mydomain.com
Domain	greendomain.ad	mydomain.com
Key distribution center (KDC), administrative server, domain controller	mydc	mydc
Realm name	GREENDOMAIN.AD	MYDOMAIN.COM

Enabling Single Sign-on for OO Central using Windows AD

Notes:

- Be sure to see the **Note** in Enabling single sign-on for OO Central for information on what is required for a user to start a flow from outside Central or to perform any actions within Central.
- The following procedure uses the ktpass tool.
- In the following procedure, the OO home directory is represented as “OO_HOME” in discussion and in commands.

To enable single sign-on for OO Central using Windows AD

1. On the domain controller (DC), in **Active Directory Users and Computers**, add an AD user account for the host (the OO Central server), using the following format:

   ```
   <service_name>/<server_name.realm.com>
   ```

OR

```
<service_name>/<server_name.realm.ad>
```

Using our example, the new account would be:

```
HTTP/mycentral.greendomain.ad
```

2. Specify the following properties for the new AD account:
   - **Password never expires**
   - **Use DES encryption types for this account**

     DES encryption is required for interoperability with Linux/Unix systems. If you do not set DES encryption types for the account, AD uses the RC4-HMAC encryption type.
   - On the **Delegate** tab, select **Trust this user for delegation to any service (Kerberos only)**.

   Note: If you are enabling SSO for Central in conjunction with a network load balancer (NLB), pay attention to the variations where they are noted (by the introductory phrase "**If you are enabling SSO for Central in conjunction with an NLB**") in the rest of this procedure.
3. On the DC, in a command-line window, generate a keytab file, using the following command:

   ```
   ktpass -out <Ctlserver_name>.keytab -princ <service_
   name>/<Ctlserver_name.domain_name>@<REALM_NAME> -mapuser <username>
   -pass *** -crypto DES-CBC-MD5 -ptype KRB5_NT_PRINCIPAL
   ```

   where:
   - `***` is the password that you specified when you created the AD account above.
   - `<Ctlserver_name>` is the name of the Central server.
   - `<username>` is the name of the AD user that you created in the previous step.

   In our example, this command would look like this:

   ```
   ktpass -out mycentral.keytab -princ
   HTTP/mycentral.greendomain.ad@GREENDOMAIN.AD
   -mapuser HTTP/mycentral.greendomain.ad -pass *** -crypto DES-CBC-MD5
   -ptype KRB5_NT_PRINCIPAL
   ```

   where `***` is the password that you specified when you created the AD account above.

   OR

   If you are enabling SSO for Central in conjunction with an NLB, you generate the keytab file for the NLB rather than for the individual Central nodes that reside behind the NLB.

   For example, suppose that:
   - The NLB machine is nlb.greendomain.ad.
   - There are two Central nodes behind the load balancer: central1.greendomain.ad and central2.greendomain.ad.

   In this case:

- The AD would be HTTP/nlb.greendomain.ad@GREENDOMAIN.AD
- The AD user account would be HTTP/nlb.greendomain.ad.
- The keytab file would be nlb.keytab.

To confirm that this command has executed successfully, find the generated keytab file in the directory that you specified in the file's fully qualified name. If you do not specify a location in the command, the keytab file is created in the directory from which you run the command.

4. Return to the properties for the new AD account and, on the **Delegate** tab, select **Trust this user for delegation to any service (Kerberos only)**.
5. Copy the keytab file (mycentral.keytab in our example) to the Central server, into the OO home directory, in the `/Central/conf` subdirectory.

   If you are enabling SSO for Central in conjunction with an NLB, you must copy the keytab file to the /Central/conf subdirectory of each Central node behind the NLB.
6. In the OO home directory, in the `/Central/conf/` subdirectory, open `jaasLogin.conf` in a text editor.

   If you are enabling SSO for Central in conjunction with an NLB, you modify the `jaasLogin.conf` (as described in the next step) on each Central node behind the NLB.
7. In `jaasLogin.conf`, following the DharmaKrb5JAAS section, add the following "com.sun.security.jgss.accept" section.

```
com.sun.security.jgss.accept {

   com.sun.security.auth.module.Krb5LoginModule

   required

   storeKey=true

   doNotPrompt=true

   useKeyTab=true

   kdc=mydc.greendomain.ad

   keyTab="<keytab_location>/mycentral.keytab"

   realm="GREENDOMAIN.AD"

   principal="HTTP/mycentral.greendomain.ad@GREENDOMAIN.AD"

   debug=true;

};
```

   where `<keytab_location>` is the path to mycentral.keytab.
8. In `Central/conf`, create a krb5.conf file that includes a definition of the default realm and KDC (or make sure that the existing krb5.conf includes that information).

   In our example, a minimal `krb5.conf` file would look like this:

```
[libdefaults]

   default_realm = GREENDOMAIN.AD
```

```
    ticket_lifetime = 24000
[realms]
 GREENDOMAIN.AD = {
   kdc = mydc.greendomain.ad
   admin_server = mydc.greendomain.ad
   default_domain = .greendomain.ad
 }
[domain_realm]
 .greendomain.ad = GREENDOMAIN.AD
 greendomain.ad = GREENDOMAIN.AD
[pam]
 debug = true
```

9. Log in to Central (with an account that in OO is a member of the ADMINISTRATOR group) and, on the **Administration** tab, click the **System Configuration** subtab.
10. Scroll down to **Kerberos Authentication Settings** and configure the location for the Kerberos 5 configuration file (`krb5.conf`) to point to `/Central/conf/krb5.conf`.

    Notes:

    - Do not set a realm or a KDC in this section, because Central is now configured to obtain them from the krb5.conf file in `/Central/conf/`. Removing the default realm or KDC is optional, for the same reason.
    - You do not need to enable Kerberos authentication unless it is used for logging in.
11. Save your changes, and then restart the Central service.
12. If the AD DC is a Windows 2000/2003 system, add the following key in its registry:

    ```
    HKEY_LOCAL_
    MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters

    Value Name: allowtgtsessionkey

    Value Type: REG_DWORD

    Value: 0x01
    ```

13. If the Central server is a Windows 2000/2003 system, add the same registry entry there.
14. To obtain a forwardable ticket:

    - To start the flow with SSO from a Linux machine that is not configured to obtain Kerberos tickets automatically, obtain a forwardable ticket from the KDC (you might have to change `/Central/conf/krb5.conf` to point it to the KDC mydc@MYDOMAIN.COM in our example), using a command like the following:

      ```
      kinit -f <sso_user>@MYDOMAIN.COM
      ```

      where `<sso_user>` is the user account that you created in step 1.

OR

- To start the flow with SSO from a Windows machine, obtain a forwardable ticket from the KDC, using the kinit executable in the `/jre1.6/bin` subdirectory of the OO home directory.

Next, you enable flows to be started from outside and/or within the Central web application.

- To enable flows to be started from outside the Central web application, see Enabling SSO for starting flows from outside Central.

Then, to start flows in SSO mode from outside Central, see Starting flows with SSO from outside Central.

- To enable flows to be started from within the Central web application, see Enabling SSO for accessing Central through the Central Web application.

Enabling Single Sign-on Using MIT KDC

Notes:

- Be sure to see the **Note** in Enabling single sign-on for OO Central for information on what is required for a user to start a flow from outside Central or to perform any actions within Central.
- The following procedure assumes that the key distribution center (KDC) is using MIT KDC.
- The OO home directory may be represented as “OO_HOME” in discussion and in commands.

To enable single sign-on using MIT KDC

1. Add a Kerberos user for each account for which you are enabling SSO.

   In our example, this user is “moe@mydc.greendomain.com”.

2. On the Central server, in the OO home directory, in the `Central/conf` subdirectory, create a `krb5.conf` file that includes definition of the default realm and KDC.

   OR

   If there is already a `krb5.conf` file there, make sure that it includes that information.

   In our example, a minimal `krb5.conf` file would look like the following:

```
[libdefaults]
        default_realm = MYDOMAIN.COM
        ticket_lifetime = 24000
        default_tkt_enctypes =  des3-cbc-sha1
[realms]
    MYDOMAIN.COM = {
        kdc = kdc.mydomain.com
        admin_server = kdc.mydomain.com
        default_domain = mydomain.com
```

```
    }


[domain_realm]

    .mydomain.com = MYDOMAIN.COM

    mydomain.com = MYDOMAIN.COM


[pam]

    debug = true
```

Note: Alternatively, you can make a copy of `sso_invoke_krb5.conf.sample`, which exists in this directory by default, rename the copy to `sso_invoke_krb5.conf`, and then edit the latter to match your domain, realm, and KDC.

3. On the KDC, create a service account for the Central server.

   Note: If you are enabling SSO for Central in conjunction with a network load balancer (NLB), pay attention to the variations in the next four steps.

4. On the KDC, add a service principal for the OO Central server (in our example, `HTTP/mycentral.mydomain.com@MYDOMAIN.COM`) using the kadmin's `addprinc` command:

   ```
   kadmin: addprinc –randkey HTTP/mycentral.mydomain.com@MYDOMAIN.COM
   ```

   OR

   If you are enabling SSO for Central in conjunction with an NLB, you create the service principal for the NLB rather than for the individual Central nodes that reside behind the NLB.

   For example, suppose that:

   - The NLB machine is nlb.mydomain.com.
   - There are two Central nodes behind the load balancer: central1.mydomain.com and central2.mydomain.com.

   In this case, the service principal would be HTTP/nlb.mydomain.com@MYDOMAIN.COM.

5. Export the principal you just created to mycentral.keytab:

   ```
   kadmin: ktadd –k mycentral.keytab HTTP/mycentral.mydomain.com
   ```

   OR

   If you are enabling SSO for Central in conjunction with an NLB, you generate the keytab file for the NLB rather than for the individual Central nodes that reside behind the NLB.

   Using the example for the preceding step, the keytab file would be nlb.keytab.

6. Copy the keytab file to the Central server, in the OO home directory, in `/Central/conf`.

   If you are enabling SSO for Central in conjunction with an NLB, you must copy the keytab file to the `/Central/conf` subdirectory of each Central node behind the NLB.

7. In the OO home directory, in the `/Central/conf/` subdirectory, open `jaasLogin.conf` in a text editor.

   If you are enabling SSO for Central in conjunction with an NLB, you modify the `jaasLogin.conf` (as described in the next step) on each Central node behind the NLB.

8. In `jaasLogin.conf`, following the DharmaKrb5JAAS section, add the following "com.sun.security.jgss.accept" section.

```
com.sun.security.jgss.accept {
        com.sun.security.auth.module.Krb5LoginModule
         required
         storeKey=true
         doNotPrompt=true
         useKeyTab=true
         kdc=kdc.mydomain.com
         keyTab="<keytab_location>/mycentral.keytab"
         realm="MYDOMAIN.COM"
         principal="HTTP/mycentral.mydomain.com@MYDOMAIN.COM"
         debug=true;
};
```

   where `<keytab_location>` is the path to mycentral.keytab.

9. Log in to Central (with an account that is a member of the ADMINISTRATOR group) and, on the **Administration** tab, click the **System Configuration** subtab.
10. Scroll down to **Kerberos Authentication Settings** and configure the location for the Kerberos 5 configuration file (`krb5.conf`) to point to `/Central/conf/krb5.conf`.

    Notes:

    - Do not set a realm or a KDC on that page, because Central will now obtain them from the `krb5.conf` file.
    - You do not need to enable Kerberos authentication unless it is used for logging in.

11. Save your changes, and then restart Central.
12. In the Central/tools subdirectory, create a jaas.conf file that looks like the following:

```
com.sun.security.jgss.initiate {
        com.sun.security.auth.module.Krb5LoginModule
         required
         client=TRUE
         useTicketCache="true";
};
```

```
com.sun.security.jgss.accept {
        com.sun.security.auth.module.Krb5LoginModule
         required
         client=TRUE
         useTicketCache="true";
};
```

The Central/tools subdirectory of the OO home directory contains the sso_invoke.sh file. This file shows how to use the Java Flow Invocation tool in single sign-on mode. You can run those shell scripts from that location.

Or, to use the invocation tool from a different machine than the Central server:

a. Copy `JRSFlowInvoke.jar`, `sso_invoke.sh`, and `sso_invoke_krb5.conf` to that machine.

b. Adjust the paths accordingly, including the path to JRE 1.6, which is required on the target machine.

You can obtain JRE 1.6 from the downloads page of the Java site.

13. To obtain a forwardable ticket:

- If the SSO flow invocation is from a Linux machine that is not configured to obtain Kerberos tickets automatically, obtain a forwardable ticket from the KDC (you might have to change `/etc/krb5.conf` to point it to the kdc.mydomain.com in our example), using a command like the following:

  ```
  kinit -f <sso_user>@MYDOMAIN.COM
  ```

  where `<sso_user>` is the user (jdoe) that you created in step 12.

  OR

- If the SSO flow invocation is from a Windows machine, obtain a forwardable ticket from the Linux MIT KDC, using the kinit executable in the `/jre1.6/bin` subdirectory of the OO home directory.

14. To give HEADLESS_FLOWS capability to the SSO users, log in to Central with an account that has Administrator rights, and then:

15. On the **Administration** tab, click the **System Configuration** subtab.

16. Scroll to the **Kerberos** section and set the default group to a group that has HEADLESS_ FLOWS capability.

    This way, any headless invocation using SSO will have the capabilities of that group. Flows cannot be invoked using the headless tool unless the user under whose credentials they are invoked has HEADLESS_FLOWS capability.

    Or, to control SSO flow invocations on a user-by-user basis:

a. On the Administration tab, create the Central user that matches the account under which the SSO flow invocation will happen ("jdoe" in our example) and specify that it is an external user.

   The group that the user is a member of must have HEADLESS_FLOWS capability; without this capability, the user cannot start runs using SSO flow invocation.

   For information on creating an external user, assigning the user to a group, and specifying a group's capabilities, see Help for Central.

   In addition to having the HEADLESS_FLOWS capability, the user under whose credentials the SSO flow invocation happens must have read and execute permissions for the flow and the operations that the flow uses.

   For more information on granting permissions to flows and operations see Help for Studio.

17. If the Central server is a Windows 2000/2003 system, add the following registry entry there:

```
HKEY_LOCAL_
MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters

Value Name: allowtgtsessionkey

Value Type: REG_DWORD

Value: 0x01
```

Next, you enable flows to be started from outside and/or within the Central web application.

- To enable flows to be started from outside the Central Web application, see Enabling SSO for starting flows from outside Central.
- Then, to start flows in SSO mode from outside Central, see Starting flows with SSO from outside Central.
- To enable flows to be started from within the Central web application, see Enabling SSO for accessing Central through the Central Web application.

Enabling SSO for Starting Flows from Outside Central

The following procedure assumes that you have already enabled SSO for OO Central. If you have not, see one of the following sections for the procedure for doing so:

- Enabling single sign-on for OO Central using Windows AD
- Enabling single sign-on for OO Central using MIT KDC

After you have enabled SSO for OO Central, you give HEADLESS_FLOWS capability to the OO users.

Note: Be sure to see the **Notes** in Enabling single sign-on for OO Central for information on what is required for a user to start a flow from outside Central or to perform any actions within Central.

To enable SSO for OO users starting flows from outside the Central web application

1. Log in to Central with an account that is a member of the ADMINISTRATOR group.
2. On the **Administration** tab, on the **System Configuration** sub-tab, use one of the two following methods to control which users can run flows from outside Central:
   - To enable multiple users to run flows from outside Central, scroll to the Kerberos section and set the default group to a group that has HEADLESS_FLOWS capability.

     This way, any headless invocation using SSO will have the capabilities of that group. Flows cannot be invoked using the headless tool unless the user under whose credentials they are invoked has HEADLESS_FLOWS capability.
   - To enable users to run flows from outside Central on a user-by-user basis:
     i. Create the Central user that matches the external account under which the SSO flow invocation will happen ("jdoe" in our example) and specify that it is an external user.

        For information on how to create a user and specify that it is an external user, see Help for Central.
     ii. Create an OO group or configure an existing OO group with HEADLESS_FLOWS capability.
     iii. Add the user to the group.

   In addition to having the HEADLESS_FLOWS capability, users under whose credentials the flows are started from outside Central must have read and execute permissions for the flow and the operations that the flow uses. For more information on granting permissions to flows and operations see Help for Studio.
3. You might have to change the path to `krb5.conf` to point it to the Windows domain controller.
4. In Central/tools, create a `jaas.conf` file that looks like the following:

```
com.sun.security.jgss.initiate {
        com.sun.security.auth.module.Krb5LoginModule
         required
         client=TRUE
         useTicketCache="true";
};
com.sun.security.jgss.accept {
        com.sun.security.auth.module.Krb5LoginModule
         required
         client=TRUE
         useTicketCache="true";
};
```

5. Create an `invoke.bat` or `invoke.sh` file:

- For running the Flow Invoke tool from a Windows machine, create an `invoke.bat` file that looks like the following:

```
SET JAVA_CMD="%ICONCLUDE_HOME%\jre1.6\bin\java"

SET MEM_OPTS=-Xmx512M


SET LOCATION=%CENTRAL_HOME%

SET INVOKE_JAR="%LOCATION%\tools\RSFlowInvoke.exe"

SET KRB5_CONF="%LOCATION%\conf\krb5.conf"

SET JAAS_CONF="%LOCATION%\tools\jaas.conf"


%JAVA_CMD% -jar %INVOKE_JAR% -krb5 %KRB5_CONF% -jaas %JAAS_CONF%

https://mycentral.greendomain.ad:8443/PAS/
services/http/execute/Library/myflows/test
```

  where:

  `%ICONCLUDE_HOME%` is the installation location of Operations Orchestration.

  `%CENTRAL_HOME%` is the installation location of OO Central.

  `/Library/myflows/test` is a flow.

- For running the Flow Invoke tool from a Linux machine, create an `invoke.sh` file that looks like the following:

```
SET JAVA_CMD="$ICONCLUDE_HOME\jre1.6\bin\java"

SET MEM_OPTS=-Xmx512M


SET LOCATION=$CENTRAL_HOME

SET INVOKE_JAR="$LOCATION\tools\JRSFlowInvoke.jar"

SET KRB5_CONF="$LOCATION\conf\krb5.conf"

SET JAAS_CONF="$LOCATION\tools\jaas.conf"


%JAVA_CMD% -jar %INVOKE_JAR% -krb5 %KRB5_CONF% -jaas %JAAS_CONF%

https://mycentral.mydomain.com:8443/PAS/
services/http/execute/Library/myflows/test
```

  where:

  `%ICONCLUDE_HOME%` is the installation location of Operations Orchestration.

  `%CENTRAL_HOME%` is the installation location of OO Central.

  `/Library/myflows/test is a flow.`

6. If the SSO flow invocation is from a Windows 2000/2003 system, add the following registry entry on the machine from which you will run `sso_invoke`:

```
HKEY_LOCAL_
MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters
 Value Name: allowtgtsessionkey
 Value Type: REG_DWORD
 Value: 0x01
```

Starting Flows with SSO from Outside Central

By default, under `OO_HOME/Central/tools` there is an `sso_invoke.bat` file for the Windows Central version (or `sso_invoke.sh` for the Linux Central version) that starts the Java flow invocation tool (`JRSFlowInvoke.jar`) in single sign-on mode. You can run those shell scripts from `OO_HOME/Central/tools` or from a different machine than the Central server.

Notes:

- On the Central server, you cannot start flows in SSO mode from outside Central.
- Be sure to see the **Notes** in Enabling single sign-on for OO Central for information on what is required for a user to start a flow from outside Central or to perform any actions within Central.

To start flows in SSO mode from outside Central

1. If you will run the SSO flow invocation tool (`sso_invoke.bat` or `sso_invoke.sh`) from a different machine than the Central server:
   a. Copy the following files to the machine from which you will run the tool:
      - `JRSFlowInvoke.jar`
      - `sso_invoke.bat` (or `sso_invoke.sh`)
      - `sso_invoke_krb5.conf`
   b. Adjust the paths in the files (including the path to JRE 1.6, which is required on the target machine).

      If necessary, you can obtain JRE 1.6 from the Downloads page of the Java site.
2. To start a flow with the `sso_invoke` tool:
   - If you are accessing the Central server directly, use a command such as the following:

     ```
     sso_invoke <Ctlserver>.<domain_name>:8443 /Library/<flow_
     path>/<flow_name>
     ```

     where:

     `<Ctlserver>` is the name of the Central server.

     `<domain_name>` is the name of the domain.

     `<flow_path>` is the path to the flow, starting below the Library folder.

     `<flow_name>` is the name of the flow.

- If you are accessing the Central server through a network load balancer, use a command such as the following:

  ```
  sso_invoke <NetwkLB>.<domain_name>:<port> /Library/<flow_
  path>/<flow_name>
  ```

  where:

  `<NetwkLB>` is the name of the network load balancer.

  `<domain_name>` is the name of the domain.

  `<port>` is the number of the port on which the network load balancer is listening.

  `<flow_path>` is the path to the flow, starting below the Library folder.

  `<flow_name>` is the name of the flow.

3. `sso_invoke.bat ctlsvrlin.mydomain.com:8443 /Library/My%20Ops%20Flows/myflow`

   Note: If the path to the flow includes spaces, you must replace the spaces with the %20 character sequence, to match the encoding of spaces in URLs.

   For instance, to start the Windows Health Check flow from outside Central in SSO mode:

   - Accessing the Central server directly, the command might look like the following:

     ```
     sso_invoke Ctlsvrwin.greendomain.ad:8443 /Library
     /Accelerater%20Packs/Operating%20Systems/Windows
     /Windows%20Health%20Check
     ```

   - Accessing Central through a network load balancer (named, say, NetwkLB), the command might look like the following:

     ```
     sso_invoke NetwkLB.greendomain.ad:8443 /Library
     /Accelerater%20Packs/Operating%20Systems/Windows
     /Windows%20Health%20Check
     ```

Enabling SSO for Accessing Central Through the Central Web Application

By default, SSO is not enabled for Internet Explorer (IE) versions 6 and 7 and Firefox. You must explicitly enable it for them.

Enabling SSO for the Web application has several configuration requirements that are completed by having previously enabled SSO for OO Central. This section's procedures for enabling SSO for the Central Web application assume as described in the following sections:

- Enabling single sign-on for OO Central using Windows AD
- Enabling single sign-on for OO Central using MIT KDC

Note: Be sure to see the note in Enabling single sign-on for OO Central for information on what is required for a user to perform any actions within Central.

To enable SSO for the Central Web application

1. To enable your Web browser for SSO:
   - For IE version 6 or 7, see the MSDN web page on "HTTP-Based Cross-Platform Authentication via the Negotiate Protocol."
   - For Firefox:
     i. Open the browser, type the following in the address bar, and then press ENTER:

     about:config

     ii. Under **Preference Name**, scroll down to and double-click the **network.negotiate-auth.delegation-uris** property.

     iii. In the **Enter string value** box that appears, enter **.greendomain.ad** and then click **OK**.

     This sets a URI pattern for the sites that are allowed SSO.

     iv. Set the same pattern for the **network.negotiate-auth.trusted-uris** property.

     v. On a Linux system, you might have to set the network.negotiate-auth.gsslib property to false if the preceding two steps are not sufficient.
2. On the Central server, in the OO home directory, navigate to `\Central\WEB-INF` and, in a text editor, open `applicationContext.xml`.
3. In the bean **authenticationEntryPoint**, in the property `loginFormUrl`, replace the value **/static/Login.htm** with **/static/SSOLogin.htm.**
4. Restart Central.
5. If the system from which you will use a browser to access the Central Web application is a Windows system, make the following changes in the registry:
   - On each Central Web application client on which Internet Explorer 8.0 is installed: If you have installed the Microsoft patch that includes "Extended Protection for Authorization," you must add to the system's Windows registry a registry entry that is specified in Microsoft Knowledge Base article 960859.
   - Add the following registry entry to the system's Windows registry:

```
HKEY_LOCAL_
MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters

Value Name: allowtgtsessionkey

Value Type: REG_DWORD

Value: 0x01
```

6. To test the configuration:
   a. On a machine other than the Central server, log in as a user that has an account in the Central server's domain (GREENDOMAIN, in these examples).

   Note: If you try this test on the Central server, it will not work.

   b. Open a browser for which you have enabled SSO (with the preceding steps in this procedure).

   c. Try to access the Central Web application, using the following syntax:

`https://<centralservername>.<domain_name>:<port>/PAS/app`

where

`<centralservername>` is the name of the Central server. The machine name that you use must exactly match the machine-name portion (in our example, mycentral) of the service account.

`<domain_name>` is name of the domain in which the Central server resides.

`<port>` is the port number over that the Central server uses to communicate. By default, this is 8443.

In our example, the following would be the URL:

https://ctlsvrlin.greendomain.com:8443/PAS/app

If SSO works for the user, the user account that is logged onto the machine from which it accesses Central will be logged in to the Central Web application without being prompted for credentials. If the SSO login fails, the standard login (username/password) screen is presented for gaining access to the Central Web application.

Among other causes, SSO might fail because the user either has no access (his or her account may have been disabled) or is not a member of an OO group.

Lightweight SSO for Operations Orchestration

Using SSO (LWSSO), you can configure OO Central to automatically log on with the same authentication information that users entered when they logged onto another HP product web client, such as the SM web client or the BSM web client. LWSSO shares a cookie between HP products web tiers that are accessed from a web browser. As a result, after logging in on another HP product that has LWSSO enabled, users enter the OO Central application directly, bypassing the OO Central logon screen.

LWSSO provides all the advantages of SSO, which are discussed in Enabling single sign-on for OO Central. You might want to use LWSSO because it is somewhat easier to enable and configure than SSO.

In a trusted, LWSSO-enabled scenario, the OO central server grants access to clients under the following conditions:

- OO Central is accessed from within another HP product web interface.
- The user's logon credentials on the HP product match those of an existing external-user account in OO.

Enabling Lightweight SSO on OO Central

Notes:

- The LWSSO feature must be enabled on all the HP products between which you want to accomplish single sign-on.
- The LWSSO enabling procedure is different on other HP products. Please consult the corresponding documentation for each HP product.
- The user account that employs SSO or LWSSO must be an external OO user (that is, not created within OO Central), and must be the same (external) user account used to log in on all HP products that use SSO or LWSSO. For information on creating an external user in OO Central, see the *OO Central User Guide*.

In the following procedure for enabling LightweightSSO on Operations Orchestration, all steps are required and must be executed in the order they are presented.

To enable LightweightSSO on OO Central

1. Stop the OO Central service (RSCentral or Central.sh).
2. Navigate to the folder `<oo installation path>\Central\WEB-INF` and open the file `applicationContext.xml` in a text editor.
3. Search for the following string:

```
<!--  LWSSO_SECTION_BEGIN

  <import resource="CentralLWSSOBeans.xml"/>

LWSSO_SECTION_END -->
```

4. Enable the import between `LWSSO_SECTION_BEGIN` and `LWSSO_SECTION_END` by adding comments to the first and third line. The new configuration should look like the example below:

```
<!--  LWSSO_SECTION_BEGIN -->

  <import resource="CentralLWSSOBeans.xml"/>

<!-- LWSSO_SECTION_END -->
```

5. Save the file.
6. Navigate to the folder `<oo installation path>\Central\WEB-INF` and open the file `web.xml` in a text editor.
7. Search for the following string:

```
<!--LWSSO_SECTION_BEGIN
```

8. Comment the line by adding an enclosing comment tag. The new line should look like the example below:

```
<!--LWSSO_SECTION_BEGIN -->
```

9. Search for the following string:

```
LWSSO_SECTION_END -->
```

10. Comment the line by adding a starting comment tag.

    The new line should look like the example below:

    ```
    <!-- LWSSO_SECTION_END -->
    ```

    There are two sequences that start with `(LWSSO_SECTION_BEGIN` and end with `LWSSO_SECTION_END)`. All instances must be commented in order to enable all the filters and mappings between them.

11. Repeat steps 7 to 10 for each of these sequences.
12. Save the file.
13. Navigate to the folder `<oo installation path>\Central\conf` and open the file `lwssofmconf.xml`in a text editor.
14. Find the following string:

    ```
    <domain>ssorealm.com</domain>
    ```

15. Replace the domain value with the domain name portion of your integrated HP product server.

    For example, if your server's fully qualified domain name is

    ```
    host.domain.hp.com
    ```

    The domain portion is

    ```
    domain.hp.com
    ```

    The new value should appear similar to the following example.

    ```
    <domain>domain.hp.com</domain>
    ```

    There are two instances of the string `<domain>ssorealm.com</domain>`

16. Repeat steps 14 and 15 for each of the instances.

    Both of them should contain the domain-name portion of your integrated HP product server.

17. Search for the following string:

    ```
    <crypto cipherType="symmetricBlockCipher" engineName="AES"
    paddingModeName="CBC" keySize="256" encodingMode="Base64Url"
    initString="CENTRAL_LWSSO_PASSPHRASE" />
    ```

18. In that string, change the `initString` value to some password you want to use to connect HP products. For example, you could specify "A1b2C3d4E5f6". With the new value inserted, the string should be similar to the following. It is recommended that the `initString` value be at least 12 characters.

    ```
    <crypto cipherType="symmetricBlockCipher" engineName="AES"
    paddingModeName="CBC" keySize="256" encodingMode="Base64Url"
    initString="A1b2C3d4E5f6" />
    ```

    Note: All HP products using LightweightSSO must use the same `initString` value.

    There are two instances of `initString`. Both of them should have the same value as the `initString` on the integrated HP product.

19. Repeat steps 17 and 18 for the second instance of `initString`.
20. Search for the following sequence:

```
<protectedDomains>
  <url>example.com</url>
</protectedDomains>
```

If you cannot find the `<protectedDomains></protectedDomains>` tags in your file, add them manually between the `<webui></webui>` tags, after the `<creation></creation>` tags. After you add the `<protectedDomains></protectedDomains>` tags, your `<webui></webui>` configuration should look similar to the example below:

```
<webui>
  <validation>
    <in-ui-lwsso>
      <lwssoValidation id="ID000001">
        <domain>domain.hp.com</domain>
        <crypto cipherType="symmetricBlockCipher" engineName="AES"
paddingModeName="CBC" keySize="256" encodingMode="Base64Url"
initString="CENTRAL_LWSSO_PASSPHRASE" />
      </lwssoValidation>
    </in-ui-lwsso>
  </validation>
  <creation>
    <lwssoCreationRef id="ID000002">
      <lwssoValidationRef refid="ID000001" />
      <expirationPeriod>600000</expirationPeriod>
    </lwssoCreationRef>
  </creation>
  <protectedDomains>
    <url>domain.hp.com</url>
  </protectedDomains>
</webui>
```

21. Replace the domain value with the domain name portion of your integrated HP product server.

For example, if your server's fully qualified domain name is `host.domain.hp.com`, the `domain` portion is `domain.hp.com`. The new value should be similar to the following example:

```
<domain>domain.hp.com</domain>
```

22. If you wish, you can add other protected domains within the `<protectedDomains> </protectedDomains>` tags.
23. Save the file.
24. Restart the Central service (`RSCentral` or `Central.sh`) for the configuration to take effect.

Enabling and Disabling Run-Scheduling Concurrency for Scheduler

You can have multiple runs of the same flow running at the same time. This means that you can start multiple runs of the same flow and target them to different servers, scheduling them to all start at the same time or to start a second run of the flow before the first one ends.

Suppose, however, that you schedule a flow such as a health check, to run twice against the same server, separating the two flows by a certain period of time. If one of the runs goes beyond the start time of the health check's next scheduled run, then the execution of the second run can interfere with the execution of the flow in the first run.

Notes:

- Because it is possible to schedule concurrent runs of flows, you need be aware of the possible interactions between concurrent runs of a flow.
- In some situations, you may wish to disable run-scheduling concurrency (by default, this capacity is enabled). You do so in the .properties file.
- When you enable or disable run-scheduling concurrency, any existing schedules are not affected. That is, any schedules that you create to run concurrently continue to be able to run concurrently without blocking each other even after you have disabled the capacity.

To enable/disable run-scheduling concurrency

1. In the Central Web application, on the **Administration** tab, click the **System Configuration** subtab.
2. Scroll down to the **Scheduler Configuration** box and locate the **If schedules of the same flow are allowed to run in parallel** setting.

   The default value for the setting is **true**, so by default, run-scheduling concurrency is enabled.
3. To disable the capacity to schedule concurrent runs of the same flow, change the setting to **false**.

   OR

   If the capacity is currently disabled and you want to enable it, change **false** to **true**.

Sizing Central for your Workload

How many concurrent runs are allowed and how much memory is reserved for the Central process determines how Central performs under your workload of flow runs.

The following considerations significantly affect your sizing strategies for Central:

- Configuring the maximum allowed concurrent runs
- Specifying the maximum RAM allowed for the Central server process
- Configuring the maximum allowed number of threads used by parallel steps

Considerations for sizing Central

The maximum addressable memory for the java process that runs the Central application is dictated by the processor architecture (32-bit or 64-bit) and by the operating system in use and its configuration. For 64-bit architectures, this addressable memory is large enough to be discounted when sizing Central. But for 32-bit architecture, the maximum addressable memory bears a significant role, as follows: The operating system may reserve a portion of either 1GB or 2GB (typically 2GB) for its own needs, which leaves the java process with an address space of 3GB or 2GB. Within this space, the main memory allocations to consider for sizing are:

- The Java Runtime Environment (JRE) itself (C1)
- Java virtual machine (JVM) bookkeeping for Java classes (C2)
- Central's heap (C3)

When planning your allocation of the available memory, consider the following:

- The sum total of these areas (C1, C2, and C3) is fixed, as explained above. Thus, an increase in one leads to a decrease in the others.
- The sizing of each memory allocation should reflect the workload planned for the Central server. Be sure to estimate the expected workload before modifying Central configuration.

By default, Central is configured with the following settings:

- maxConcurrentRuns = 600

  This setting specifies how many threads are available for running flows.
- For 32-bit OSs, the Central maximum heap size is 768MB
- For 64-bit OSs, the Central maximum heap size is 1024MB

Note: The 64-bit settings are conservative, and the default 64-bit settings are relatively close to the 32-bit settings. It is recommended that after deploying Central, you reconfigure these settings for 64-bit installations, based on available memory and intended usage.

These settings are based on average observed workloads. An average flow is considered to consist of at most a couple of hundred steps, processing small pieces of text—that is, less than 10 kilobytes per step.

Before increasing the value of the maxConcurrentRuns setting, consider carefully the size of the planned workload. When you estimate the workload, consider the following contributing factors:

- Size of the flows in steps.

  Large flows (having more than, say, 500 steps) require more heap memory than shorter ones.

- The amount of data that the flows process.

  A flow that processes large pieces of text generally uses significantly more heap memory than a flow that processes smaller (on the order several kilobytes) pieces of text.

- The interface used to run the flows.

  The amount of heap memory required to run a flow from Central's Web UI increases in direct proportion to the number of steps in the flow.

  In contrast, the heap memory required to run a flow via a command-line tool (such as the flow invoker utility JRSFlowInvoke) is constant—that is, it does not increase with the number of steps in the flow. The amount of heap memory required for a flow in the Web UI can reach hundreds of megabytes for flows that have tens of thousands of steps, whereas the same flow run through JRSFlowInvoke requires less than 10 megabytes.

- How many users are concurrently logged in to the Web UI.

  The heap memory usage associated with keeping flow repository data for each user can reach as much as 50MB per user.

- Running flows and viewing reports increases the heap memory requirements described earlier in this list.

In addition to the estimated expected workload, keep in mind the following uses of memory:

- For 32-bit operating systems, the Central process is limited to spawning around 1200 threads on Windows and around 1500 on Linux. This is a limitation related to the amount of memory addressable by a 32-bit process and the amount of memory required for the stack of each thread. Memory for threads is allocated from the memory required by the JRE.

- Each flow run requires at least one thread (it requires more if it has parallel steps).

  In addition, the Central application requires a number of threads for tasks other than run steps: database communication, scheduling, etc.

- Increasing the heap size of a Java process makes less memory space available for threads. In other words, the larger the heap size, the fewer threads can be created.

  Experiments show that for a 32-bit Central process to be able to create 1200 concurrent flow runs, the heap size should be around 512 megabytes (MB) for the current version of the Java™ Virtual Machines. Thus, larger configurations require 64-bit servers.

Considering all the above, there are two extremes in the range of configurations for sizing Central:

- One is to run as many flows as possible by allocating the minimum possible size for the heap and a maximum of 1200 (or 1500) concurrent runs on a 32-bit operating system. This

configuration would mandate running flows invoked through command-line tools (which do not require large amounts of memory to support a UI).

- The other is designed to accommodate as many concurrent users as possible on the Central Web UI. This configuration would maximize the amount of heap at the expense of the number of allowed concurrent runs.

The default configuration shipped for 32-bit systems strives to achieve middle ground: a reasonable amount of concurrent runs with a reasonable amount of heap size to accommodate Web UI users.

For 64-bit systems, the correlation between maxConcurrentRuns and the amount of heap is less stringent and is largely dependent upon the amount of available RAM for the Central server.

To increase the heap size, you edit the wrapper configuration file. For information on doing so, see Specifying the maximum RAM allowed for the Central server process, below.

Configuration for workloads that include parallel steps

If the workload is expected to run flows that have parallel-processing steps (that is, parallel split steps, multi-instance steps, or nonblocking steps), then you must consider an additional parameter: the number of total threads that can be used by such parallel steps. This configuration is given by the parameter dharma.runengine.parallel.max.threads. You add this parameter's value to the value for maxConcurrentRuns to determine the total number of threads used solely for the purpose of running flows.

For example, assume the expected workload contains a maximum of 600 concurrent flow runs. Further, assume that each of these runs is expected to execute a multi-instance step that spawns 5 parallel operation executions. If each flow run was able to execute at full concurrency, the total maximum number of threads required from the system would be 600 + (600 * 5) = 3600 threads. This concurrency cannot be supported on a 32-bit system. By setting the parameter dharma.runengine.parallel.max.threads to 300, the peak concurrency is limited to 600 + 300 = 900 threads. This parameter limits the size of the thread pool used for the execution of parallel step operations (which includes multi-instance steps, parallel split lanes, and nonblocking steps). With this configuration, some of the multi-instance steps in the example may achieve a concurrency of 5 threads, depending on their timing, whereas others may not achieve any concurrency at all and may run all of their 5 parallel step operations in the same parent run thread (the run is effectively serialized).

Configuring the maximum allowed concurrent runs

By default, the maximum number of runs that the Central server can run at the same time is 600. If you choose to increase this, you may need also to increase the maximum amount of RAM allotted to the OO server's Central process. For information on doing so, see Specifying the maximum RAM allowed for the Central server process. Remember also that increasing the number of runs that Central can execute simultaneously beyond the capabilities of your system, can affect performance.

To configure the maximum number of concurrent runs

1. In the OO home directory, navigate to the `\Central\conf subdirectory`, and open Central.properties.
2. Find the following line

   ```
   dharma.executor.maxConcurrentRuns=600
   ```

3. Change `600` to the greatest number of runs that your system should run at one time.
4. Save and close the file.

Configuring the Maximum Allowed Number of Threads Used by Parallel Steps

By default, the maximum number of threads that the Central server can use for running parallel steps is 300. If you choose to increase this, you may need to also increase the maximum amount of RAM allotted to the OO server's Central process. For information on doing so, see Considerations for Sizing Central.

To configure the maximum number of threads used by parallel steps:

1. In the OO home directory, navigate to the `\Central\conf` subdirectory, and open the file named `Central.properties`.
2. Find the following line:

   ```
   dharma.runengine.parallel.max.runs=300
   ```

3. Change 300 to the greatest number of threads you wish to allow for the execution of parallel steps at any one time.
4. Save and close the file.

Specifying the Maximum RAM Allowed for the Central Server Process

To specify the maximum RAM available for the Central process

1. In the OO home directory, in the `\Central\conf\` subdirectory, open `wrapper.conf`.

   Note: In Linux, the file in Central that needs to be edited is called `wrapper_linux.conf`.

2. Find the following line:

   `wrapper.java.maxmemory=768`

3. Change 768 to a value that represents a desirable maximum amount of memory for the service.

   In `wrapper.conf`, you can leave 'm' off the end of the value, because this value always represents megabytes.

4. Save and close the file.

Changing the Timeout Limit for RAS Operations

RAS operations are subject to a default timeout limit of 20 minutes on Central for remote RAS operations. To support RAS operations that are likely to take more than 20 minutes to complete, you can change Central's default timeout setting.

To change the timeout setting for a remote RAS installation

1. In the OO home directory, navigate to `\Central\conf\` and open the `wrapper.conf` file for editing.

   > **Note:** In Linux, the file in Central that needs to be edited is called `wrapper_linux.conf`.

2. Add the following line to the file:

   `wrapper.java.additional.<n>=-Dras.client.timeout=<timeout in seconds>`

   Where:

   - `<timeout in seconds>` is the amount of time you want the RAS to wait before timing out.
   - `<n>` is the next increment for the java additional parameters configured through this file.

   This line overrides the default timeout value for RAS operations.

SSH operations also have their own timeout settings of 90 seconds. When you use a (copy of a) SSH operation from the Studio Library **Operations** folder, you can change the timeout setting by changing the value for the **Timeout** input of the operation.

Changing Studio Configurations

In the OO home directory, in the `\Studio\conf\` subdirectory, in the `Studio.properties` file, you can change the following aspects of Studio:

- Central host server
- Default communication port used
- Choice of HTTP: or HTTPS: (secure sockets) as the Internet protocol
- Whether the password will be erased when a repository containing system accounts is exported

To change the Studio.properties file

1. In the OO home directory, open the `\Studio\conf\` subdirectory, and then with a text editor, open `Studio.properties`.
2. Edit the following lines to make the desired changes:
   - To change the name of the host server (the server on which the Web application is located), change `localhost` in the following line to the name or IP address of the host server.

     ```
     dharma.repaircenter.host=localhost
     ```

   - To change the port number that OO uses, change 8443 in the following line to the desired port number.

     ```
     dharma.repaircenter.port=8443
     ```

   - By default, OO uses the https Internet protocol. To specify that OO use the http Internet protocol, change `https` to `http`in the following line.

     ```
     dharma.repaircenter.proto=https
     ```

   - To prevent the passport from being erased when a repository containing system accounts is exported, add the following line:

     ```
     dharma.repo.allow.system.accounts.travelling=true
     ```

     This line also needs to be added to the `Central.properties` file. For more information, see "Preventing System Account Password Reset When Exporting a Repository" on page 70

Changing Anti-Samy Input Filter Settings

Anti-Samy is an HTML, CSS and JavaScript filtering system for Java that protects your system from XSS attacks. Anti-Samy filters sanitize user input that is displayed in the Central web application. The level of filtering is based on which of the available policies is specified. Anti-Samy is always enabled in the Central web application; by default, the policy used is slashdot.xml. You can specify that Anti-Samy use one of the five following policies:

- `ebay.xml`

  A policy that allows rich HTML.

- `myspace.xml`

  Allows any HTML and CSS elements that do not include JavaScript.

- `slashdot.xml`

  One of the most restrictive policies. This policy does not allow CSS, and allows strict text formatting, that is, the following HTML tags.

  - (to make text bold)
  - <u></u> (to underline text)
  - (to make text italic)
  - <a> (the anchor tag, used to create references)
  - <blockquote></blockquote> (to create block quotes)

- `tinymce.xml`

  This policy allows only text formatting.

- `anythinggoes.xml`

  Allows all HTML, CSS, and JavaScript elements. This policy provides the least amount of protection against external attacks. It is strongly recommended that you not select this policy unless you trust the inputs supplied to the Central web application.

To change the Anti-Samy policy

1. In the OO home directory, in the `\Central\conf\` subdirectory, open the `Central.properties` file.
2. Find the line `dharma.antisamy.default.policy=slashdot.xml`.
3. To change the policy that Anti-Samy uses, type one of the following:
   - ebay.xml
   - myspace.xml
   - slashdot.xml

- tinymce.xml
- anythinggoes.xml

4. Save and close the file.
5. Stop and restart the RSCentral service.

For more information on Anti-Samy and the policies see the Anti-Samy Project web site.

Preventing System Account Password Reset When Exporting a Repository

By default, when a repository containing system accounts is exported, the password is erased. In order to prevent this from happening, you can add a line to the `Central.properties` and `Studio.properties` files.

To stop the password from being erased when a repository containing system accounts is exported

1. In the OO home directory, navigate to the `\Central\conf` subdirectory, and open the `Central.properties` file with a text editor.
2. Add the following line:

   ```
   dharma.repo.allow.system.accounts.travelling=true
   ```

3. Save and close the file.
4. Repeat the process for the `Studio.properties` file, in the `\Studio\conf\` subdirectory.

Configuring Log File Settings

OO records errors (ERROR), warnings (WARN), information (INFO), and debugging messages (DEBUG) in the following log files:

- For Studio: `Studio.log`, in the `\Studio\logs` subdirectory of the OO home directory
- For Central: `Central_wrapper.log`, in the `\Central\logs` subdirectory of the OO home directory
- For the Web Service Wizard

Because logging activity can slow OO's performance and create very large log files, it is important that OO run with appropriate logging levels. The default logging levels have been set to provide necessary information without impacting performance. It is recommended that you use the default logging levels.

To change logging levels

1. In the `jetty\resources` subdirectory of the OO home directory, modify the `log4j.properties` file according to your needs.
2. Save changes.

Auditing Security-related Application Events

You should set up the `log4j.properties` file to log security-related application events. Log entries should identify the individual whose action is being audited, the individual affected by the action, and the time of the action. This enables easier detection of security threats.

Security-related application events include:

- Failed authentication / login attempts
- Accesses with insufficient user rights

1. To activate the auditing, use the `log4j.properties` file to log the audit data.The log level is set by default to INFO.

   The audit log file is defaulted to (`%ICONCLUDE_HOME%\Central\logs\Central_audit.log` for Windows systems or `$ICONCLUDE_HOME\Central\logs\Central_audit.log` for Linux systems), which is local on the node. It can also be directed to a database.

2. Log the following types of events:
   - **LOGIN_OK**
   - **LOGIN_FAILED**
   - **LOGOUT**

- **PERMISSION_DENIED** – when a user tries to access a section without having the rights. This includes:
  - Studio – no AUTHOR capability
  - headless requests – no HEADLESS_FLOWS
  - Central scheduler – no SCHEDULE or VIEW_SCHEDULES
  - Central configuration – no MANAGE_USERS, MANAGE_GROUPS, or MANAGE_CONF

The structure of the logged data should be as follows:

- **timestamp** – the time in milliseconds, when the event happened
- **node** – the host (IP) where it happened (this doesn't always apply). In a cluster, the audit data from all the nodes would be aggregated.
- **event** – the type of the event
- **source host** – the remote host that made the request
- **client** – how the service was called: CENTRAL, STUDIO, PORTAL, SOAP, HEADLESS
- **user** – the user name used in the request
- **cause** – especially for failures/errors; if possible, specify the cause
- The fields are separated by ';'. If a value for a field cannot be determined, it will be logged as empty string.

Example of a Log4j configuration:

```
#
# central audit
# INFO level
#
log4j.logger.com.iconclude.dharma.util.CentralAudit=INFO,central
log4j.additivity.com.iconclude.dharma.util.CentralAudit=false
log4j.appender.central=org.apache.log4j.RollingFileAppender
log4j.appender.central.File=${iconclude.home}/Central/logs/central-audit.log
log4j.appender.central.MaxFileSize=10000KB
log4j.appender.central.MaxBackupIndex=1
log4j.appender.central.layout=org.apache.log4j.PatternLayout
log4j.appender.central.layout.ConversionPattern=%d{ISO8601} - %m%n
```

Example of a log:

```
2012-05-29 13:37:16,727 - 1338287836727;16.53.237.105;LOGIN_
FAILED;localhost:8082;Portal;admin;Failure: getAuthenticationObject
() securityService.mapAuthorities - Invalid username or password

2012-06-12 11:35:42,200 - 1339490142200;16.53.237.105;PERMISSION_
DENIED;127.0.0.1;CENTRAL;User1;"User is missing the following
capabilities: Allows reporting and viewing the dashboard"

2012-06-12 15:55:27,880 - 1339505727880;16.53.237.105;LOGIN_
OK;127.0.0.1;CENTRAL;admin;

2012-06-12 15:55:30,071 -
1339505730071;16.53.237.105;LOGOUT;127.0.0.1;CENTRAL;admin;
```

Note: Limitation: OO does not log failures in SOAP calls that happen due to lack of permissions on objects (or lack of capabilities). The audit data is logged locally in a file by each individual node and is not aggregated for the whole cluster. To overcome this limitation, one option is to log into a common database. Another could be to develop an aggregation mechanism or tool.

Backing up Operations Orchestration

Backing up OO involves backing up your flows, operations, system accounts, selection lists, and other OO objects, and backing up the OO database. You back up OO objects in Studio by backing up the repository and then placing a copy of the repository's backup in a secure location.

To back up OO

1. In Studio, back up each repository (**Create Backup** command, on the **Repository** menu), using the procedure given in Help for Studio.

   Each repository is backed up as a .zip file.

2. Make a copy of each repository's .zip file and store the copy in a secure location.
3. Back up the Central database and store the backup in a secure location.

   Dashboard charts are stored in the Central database, so the database backup includes Dashboard charts.

4. Back up the OO home directory.

Supporting a Central Server Cluster

For information on supporting a Central server cluster, see *Operations Orchestration High Availability Guide*.

Troubleshooting

Flows run under heavy load with command-line or RAS operations fail on Windows systems with error code 128

If the Central server or a standalone RAS installation has a Windows operating system, flows that run on those machines under intense usage and using command-line operations or the RAS operation may fail with error code 128. This can result from Data Execution Prevention (DEP) being enabled for all programs. By default, DEP is enabled for all programs and services except any exceptions that you specify. Alternatively, try enabling DEP “for essential Windows programs and services only.” For information on changing the DEP settings, see Help for Windows.


Java™
COMPATIBLE