

HP Operations Orchestration Software

Software Version: 7.51

Administrator's Guide

Document Release Date: August 2009

Software Release Date: August 2009



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 Notices

© Copyright 2009 Hewlett-Packard Development Company, L.P.

Trademark Notices

All marks mentioned in this document are the property of their respective owners.

Finding or updating documentation on the Web

Documentation enhancements are a continual project at Hewlett-Packard Software. You can obtain or update the OO documentation set and tutorials at any time from the HP Support web site.

To obtain HP OO documentation and tutorials

1. Go to the HP Software Product Manuals web site (<http://support.openview.hp.com/selfsolve/manuals>).
2. Log in with your HP Passport user name and password.
OR

If you do not have an HP Passport, click **New users – please register** to create an HP Passport, then return to this page and log in.

If you need help getting an HP Passport, see your HP OO contact.

3. In the **Product** list box, scroll down to and select **Operations Orchestration**.
4. In the **Product Version** list, click the version of the manuals that you're interested in.
5. In the **Operating System** list, click the relevant operating system.
6. Click the **Search** button.
7. In the **Results** list, click the link for the file that you want.

Where to find Help, tutorials, and more

The HP Operations Orchestration software (HP OO) documentation set is made up of the following:

- Help for Central
Central Help provides information to the following:
 - Finding and running flows
 - For HP OO administrators, configuring the functioning of HP OO
 - Generating and viewing the information available from the outcomes of flow runsThe Central Help system is also available as a PDF document in the HP OO home directory, in the \Central\docs subdirectory.
- Help for Studio
Studio Help instructs flow authors at varying levels of programming ability.
The Studio Help system is also available as a PDF document in the HP OO home directory, in the \Studio\docs subdirectory.
- Animated tutorials for Central and Studio
HP OO tutorials can each be completed in less than half an hour and provide basic instruction on the following:
 - In Central, finding, running, and viewing information from flows
 - In Studio, modifying flowsThe tutorials are available in the Central and Studio subdirectories of the HP OO home directory.
- Self-documentation for operations and flows in the Accelerator Packs and ITIL folders

Self-documentation is available in the descriptions of the operations and steps that are included in the flows.

Support

For support information, including patches, troubleshooting aids, support contract management, product manuals and more, visit the following site: <http://www.hp.com/go/bsaessentialsnetwork>

This is the **BSA Essentials Network** Web page. To sign in:

1. Click **Login Now**.
2. On the **HP Passport sign-in** page, enter your HP Passport user ID and password and then click **Sign-in**.
3. If you do not already have an HP Passport account, do the following:
 - a. On the **HP Passport sign-in** page, click **New user registration**.
 - b. On the **HP Passport new user registration** page, enter the required information and then click **Continue**.
 - c. On the confirmation page that opens, check your information and then click **Register**.
 - d. On the **Terms of Service** page, read the Terms of use and legal restrictions, select the **Agree** button, and then click **Submit**.
4. On the **BSA Essentials Network** page, click **Operations Orchestration Community**.
The Operations Orchestration Community page contains links to announcements, discussions, downloads, documentation, help, and support.

Note: Contact your OO contact if you have any difficulties with this process.

Table of Contents

Warranty	ii
Restricted Rights Legend	ii
Trademark Notices	ii
Finding or updating documentation on the Web.....	iii
Where to find Help, tutorials, and more	iii
Support	iv
Where administrative tasks are documented	1
Overview	1
Default ports used by HP OO components	2
Security: Users, Groups, Capabilities, and Permissions.....	2
Configuring Active Directory or LDAP over SSL (LDAPS protocol).....	3
Replacing a security certificate	5
Modifying the RAS keystore to share mutual SSL authentication with Central.....	8
Scheduler certificate replacement.....	9
Studio certificate replacement.....	9
Testing the certificates.....	9
Configuring HP OO for extended functionality	10
Changing Central configurations	11
Changing the maximum size of the log files.....	11

Enabling single sign-on for OO Central	12
Enabling single sign-on for OO Central using Windows AD.....	13
Enabling single sign-on using MIT KDC.....	16
Enabling SSO for starting flows from outside Central.....	19
Starting flows with SSO from outside Central	21
Enabling SSO for accessing Central through the Central Web application.....	23
 Enabling and disabling run-scheduling concurrency for Scheduler.....	 24
 Sizing Central for your workload.....	 25
Considerations for sizing Central	25
Configuration for workloads that include parallel steps.....	27
Configuring the maximum allowed concurrent runs	27
 Configuring the maximum allowed number of threads used by parallel steps	 27
 Specifying the maximum RAM allowed for the Central server process	 28
 Changing RAS-related configurations	 28
Enabling Java database connection pooling for certain flows.....	28
Contents of databasePooling.properties.....	29
Changing the timeout limit for RAS operations.....	34
 Changing Studio configurations	 34
 Configuring log file settings	 35
 Backing up OO.....	 35
 Supporting a Central server cluster	 35
 Troubleshooting.....	 36
Flows run under heavy load with command-line or RAS operations fail on Windows systems with error code 128.....	36
 Index	 37

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 **Administrative** tab. For example, on the Central **Administrative** 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 Central *User's Guide*).

Information on administrative tasks related to configuring and supporting a Central server cluster is in *Clustering and Load Balancing for HP Operations Orchestration 7.50* (ClusteringGuide.pdf).

This *Administrator's 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 *Clustering and Load Balancing for HP Operations Orchestration 7.50* (ClusteringGuide.pdf).

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
- Scheduler: 19443

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.
- 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.

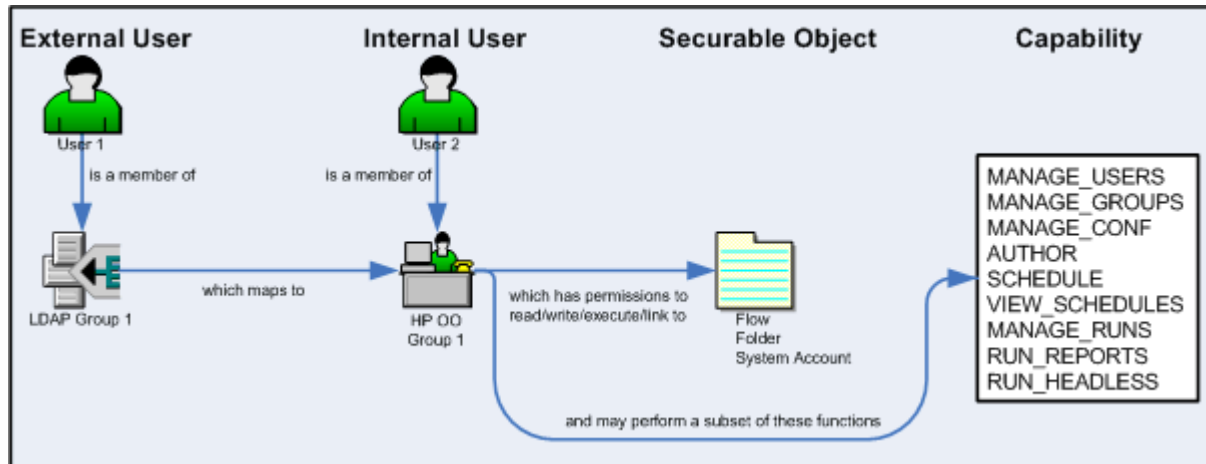


Figure 1 - Components of access control in OO

- 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 the **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 (`%OO_HOME%` on a Windows system or, on a Linux system, `$OO_home`), then to the `jre1.6` directory's `bin` subdirectory.
17. To import the certificate, run the following command:

```
keytool -importcert -keystore "%OO_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 "%OO_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 on the **Administration** tab (**System Configuration** subtab) of the OO Central web application. 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](#)
- [Modifying the RAS keystore to share mutual SSL authentication with Central](#)
- [Scheduler certificate replacement](#)
- [Studio certificate replacement](#)
- [Testing the certificates](#)

Notes:

- In the following procedures, `%OO_home%` represents the OO home directory.
- The following procedure assumes that you have OpenSSL installed.
- This procedure assumes that the certificates are in the Privacy Enhanced Mail (PEM) format. If your certificates are in Distinguished Encoding Rules (DER) format, you will modify one of the steps. The modification is described in the procedure.

To replace the default security certificate

1. Make a backup copy of the following keystores:

- `%OO_home%\Central\conf\copy rc_keystore`
- `%OO_home%\ras\Java\Default\webapp\conf\ras_keystore.jks`

You can use any pair of public/private certificates based on your specific certificate requirements, security policies, etc. This procedure includes an example of using the standard java keytool to generate a self signed certificate"

Notes: The value for CN must match the name that you will use in the URL, which is not necessarily the host name of the server.

Important Note: 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.

2. Stop the RSCentral, RSScheduler and RSJRAS services.
3. Create new keystores with public and private certificates.
4. Run the following OpenSSL command to create a pkcs12 version of the certificate (with private key attached, make sure to use the private key's password).

```
>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` so that the command is as follows:

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

5. To import the pkcs12 into a java keystore, run the following command (you may have to change the path to where Central and/or RAS is installed):

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

- Use the password above when asked for input keystore passphrase.
- Be sure to set a password for output keystore passphrase (they should be the same)

6. 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:

```
pas, 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

7. Do one of the following:

- To trust all SSL connections based on CA, import the CA certificate into the keystore.

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

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

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

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

Important Notes:

- Specifically for Central, you'll need a certificate that is purposed for both server authentication and client authentication. This will probably not be the default SSL/Web Server certificate issued by the CA.
 - When the CA Web site responds to your request, specify Download CA certificate.
 - Before importing the response into the OO keystore, import the certificates for each CA in the chain leading up to the CA that signed the certificate request.
8. To verify that the certificate that the CA issued was correctly purposed (for both server authentication and client authentication), open response.cer.
 9. On the **Details** tab, scroll down to the **Enhanced Key Usage** field.

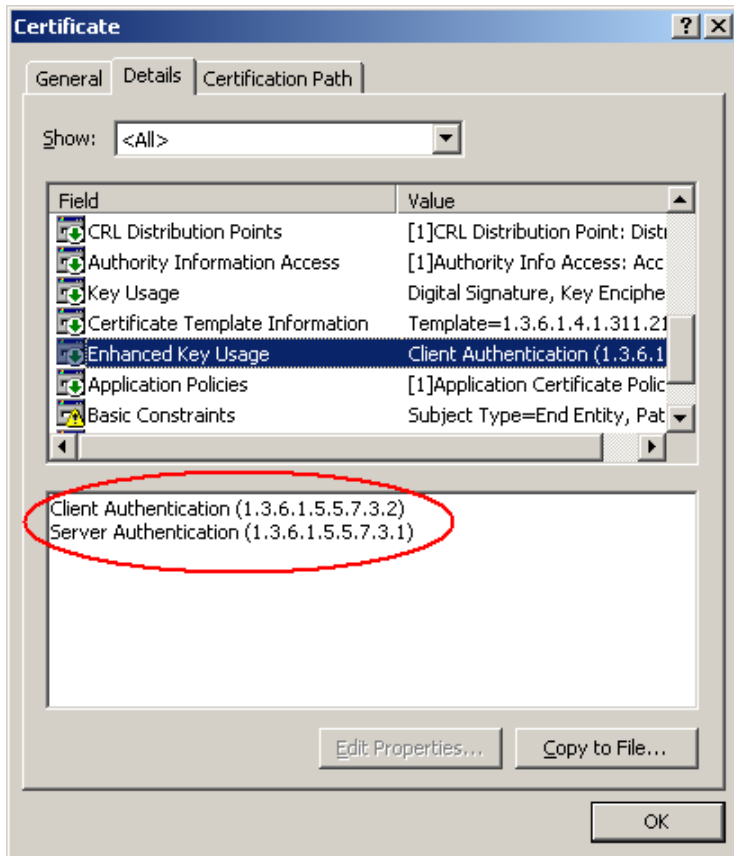


Figure 2 - response.cer editor

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

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

Make note of the `OBF:<string>` value; 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.

11. 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.

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

12. Execute the following command:

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

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

Modifying the RAS keystore to share mutual SSL authentication with Central

Note: This procedure assumes that the certificates are in the Privacy Enhanced Mail (PEM) format. If your certificates are in Distinguished Encoding Rules (DER) format, you will modify one of the steps. The modification is described in the procedure.

To modify the RAS keystore to share mutual SSL authentication with Central

1. To create a new keystore with public and private certificates:
 - a. Run the following OpenSSL command to create a pkcs12 version of the certificate (with private key attached, make sure to use the private key's password).

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

If the certificate format is DER, add the `-inform DER` parameter after `pkcs12` so that the command is as follows:

```
>openssl pkcs12 -inform DER -export -in <cert.pem> -inkey <key.key> -out ras.p12 -name ras
```
 - b. 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 Central and/or RAS is installed)

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

Use the password above when asked for input keystore passphrase.
Make sure to set a password for output keystore passphrase (they should be the same).
2. 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. Do one of the following:
 - To trust all SSL connections based on the CA, import the CA certificate into the keystore.

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

At the prompt **Trust this certificate?** type **yes**.
 - To import all Central Servers Public Certificates:

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

When asked to **Trust this certificate?** Reply yes.
4. To obfuscate the keystore and private key passwords for use in the jetty.xml file, use the following command:

```
> Java -cp "%OO_HOME%\Jetty\lib\jetty-6.1.14.jar";"%OO_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.
5. 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.
6. Start the RSCentral, RSScheduler and RSJRAS services.
- Next, you modify the Scheduler keystore to match up with the Central keystore

Scheduler certificate replacement

Scheduler needs a copy of the rc_keystore that Central has.

1. Copy rc_keystore to %OO_HOME%\Scheduler\conf\rc_keystore.
2. Update the Scheduler's secured.properties file as follows:


```
>"%OO_HOME%\jre1.6\bin\java.exe" -cp "%OO_HOME%\Central\tools\lib\secure-props.jar";"%OO_HOME%\Central\WEB-INF\lib\dharma-commons.jar" ;"%OO_HOME%\Central\WEB-INF\lib\commons-cli-1.0.jar" ;"%OO_HOME%\Central\WEB-INF\lib\ant.jar" ;"%OO_HOME%\Central\WEB-INF\lib\log4j-1.2.8.jar" -f "%OO_HOME%\Scheduler/conf/secured.properties" com.iconclude.dharma.tools.SecureProps -d dharma.security.ssl.trustStorePassword=<keystore password> -d dharma.security.ssl.keyStorePassword=<keystore_password>
```
3. Edit jetty.xml as follows:
 - Update the `<Set name="Password">OBF:<string></Set>` value with the `OBF:<string>` value for your keystore password that you recorded.
 - Update the `<Set name="KeyPassword">OBF:<string></Set>` value with the `OBF:<string>` value for your private key password that you recorded.

Next, you'll replace the Studio certificate.

Studio certificate replacement

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.

To replace Studio's certificate

- Update Studio's secured.properties file:


```
>"%OO_HOME%\jre1.6\bin\java.exe" -cp "%OO_HOME%\Central\tools\lib\secure-props.jar";"%OO_HOME%\Central\WEB-INF\lib\dharma-commons.jar" ;"%OO_HOME%\Central\WEB-INF\lib\commons-cli-1.0.jar" ;"%OO_HOME%\Central\WEB-INF\lib\ant.jar" ;"%OO_HOME%\Central\WEB-INF\lib\log4j-1.2.8.jar" -f "%OO_HOME%\Studio/conf/studio-secured.properties" com.iconclude.dharma.tools.SecureProps -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.)

There are two considerations that 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.
- 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 7.50 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.

- [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 and configuring the
- [Changing the timeout limit for RAS operations](#)

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
 \Scheduler\logs\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.

To change the maximum size of a log file

4. 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

\Central\conf\ wrapper.conf

- Scheduler
 \Scheduler\conf\ wrapper.conf
- RAS
 \RAS\Java\Default\webapp\conf\ wrapper.conf

5. 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.

Important: Setting the value to zero (0) disables rolling, and the file will grow indefinitely.

6. Save and close the file.

Enabling single sign-on for OO Central

As related to starting OO flow runs, *single sign-on (SSO)* is the ability to start a flow run from outside Central using the credentials of the account that is already logged on to the computer or to open the Central Web application without having to log in to Central separately. With either use, SSO can provide security and performance benefits.

Note: 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.

Important: SSO does not by itself enable users to do anything once signed in to Central. Any user must be a member of a 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 *Guide to Authoring Operations Orchestration Flows* (Studio_AuthorsGuide.pdf).

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](#)

Important: 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 **Important** 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 account for the host (the OO Central server), using the following format:

`<service_name>/<server_name.domain_name>`

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 in the next four steps.

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. 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.

5. 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.

6. 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>` in the highlighted line is the path to `mycentral.keytab`.

7. 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

```

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, KDC, and admin server.

8. 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.
9. 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.

10. Save your changes, and then restart the Central service.

11. 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

```

12. If the Central server is a Windows 2000/2003 system, add the same registry entry there.

13. 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 **Important** 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](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](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>` in the highlighted line 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:

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

- a. On the **Administration** tab, click the **System Configuration** sub-tab.
- b. 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.

- b. 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.

15. 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 **Important** 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.

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:
 1. 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.
 2. Create an OO group or configure an existing OO group with HEADLESS_FLOWS capability.
 3. 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="%OO_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:

%OO_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="$OO_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:

%OO_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 **Important** 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.

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, <http://java.sun.com/>).

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 ctlsvr.lin.mydomain.com:8443 /Library/My%20Ops%20Flows/myflow`

Important: 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/Accelerator%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 **Important** 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:
 1. Open the browser, type the following in the address bar, and then press ENTER:
about:config
 2. Under **Preference Name**, scroll down to and double-click the **network.negotiate-auth.delegation-uris** property.
 3. In the **Enter string value** box that appears, enter **.greendomain.ad** and click **OK**. This sets a URI pattern for the sites that are allowed SSO.
 4. Set the same pattern for the **network.negotiate-auth.trusted-uris** property.
 5. 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, 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://ctlsvr1in.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.

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.

Important: 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.

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 Scheduler's `scheduler.properties` file.

Important: 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 are considerations that significantly affect the sizing of 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](#)

There are a number of considerations to keep in mind when you decide how to size Central.

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 how to allocate 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.
Important: 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 `dharmarunengine.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 `dharmarunengine.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
`dharmarunengine.parallel.max.threads=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
`dharmarunengine.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.
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 RAS-related configurations

Enabling Java database connection pooling for certain flows

When a SQL request from one of the flows under Library/Operations/Databases/JDBC or Library/Operations/Databases/Oracle is sent to SQL.java (underlying code for handling query), the SQL request includes a request for a database connection. If you need to enable database connection pooling, carry out the following procedure.

To enable connection pooling

1. Make sure that the databasePooling.properties file is in the OO home directory, in the \RAS\Java\Default\webapp\conf subfolder.
2. If the databasePooling.properties file is not there:
 - a. Create a file in that location and name it databasePooling.properties.
 - b. Copy the material under the heading Contents of databasePooling.properties into the file.
 - c. Configure the settings in the file to fit your needs.

Among the properties to configure:

- To enable connection pooling, set the property db.pooling.enable as follows:

`db.pooling.enable=true`

In the following template for databasePooling.properties, the property's value is already set to true.

Contents of databasePooling.properties

```
# #####
# For the flows under
# Library/Operations/Databases/JDBC
# Library/Operations/Databases/Oracle
#
# NOTE:
# By default, flows support Oracle, MSSql and Sybase. To enable
# MySQL support, the JDBC drivers would need to be added to
# the %ICONCLUDE_HOME%\RAS\Java\Default\webapp\WEB-INF\lib directory
# and then update the SQLDBType selection list with the appropriate
# setting (MySQL).
# To enable Netcool support, add the JDBC driver to the
# %ICONCLUDE_HOME%\RAS\Java\Default\webapp\WEB-INF\lib directory
# and then update the SQLDBType selection list with the appropriate
# setting (Netcool). The JDBC driver is included with Netcool
# (not Operations Orchestration), and is named jconn2.jar or jconn3.jar,
# depending on the Netcool version.
#
# CUSTOM NOTE:
# If you want to use a custom JDBC driver than you must have the library
# installed as noted above. You need to entry Custom for input DbType.
# Also, you will need add two new inputs to the operation:
# dbClass - the classname of the JDBC driver to use
# dbURL - the url required to load up the driver and make your connection
#
# For Windows authentication, the domain must be specified in dbURL
# and dbClass must point to a MSSQL driver.
# For Sql authentication, the domain must not be specified in dbURL.
#
# The following configuration is for Java connection pooling.
# #####

# -----
# general configuration for all
# -----
#
# Enable connection pooling, if this set to false, rest of the
# Java connection pooling configuration will be ignored
#

db.pooling.enable=true

#
# Interval in seconds for a low priority thread to
# clean the pooled datasource that has 0 connections
```

```

# default is 2 hours
#
db.datasource.clean.interval=7200

# -----
# General configuration per DBMS
# -----

# Notes: The following <dbtype>.connection.total.maxpoolsize should
# be tied with the property connection.maxpoolsize which is per user.
#
# For example, if set connection.maxpoolsize = 20,
oracle.connection.total.maxpoolsize=100,
# then, max 5 users can connect to the database at the same time.
# Make sure connection.maxpoolsize * number of pooled datasource (per user)
# <= oracle.connection.total.maxpoolsize
# Adjust these properties carefully, per your business needs.

#
# total connections per DBMS for oracle server
#

oracle.connection.total.maxpoolsize=100

#
# total connections per DBMS for sql server
#
mssql.connection.total.maxpoolsize=100

#
# total connections per DBMS for mysql
#
mysql.connection.total.maxpoolsize=100

#
# total connections per DBMS for db2
#
db2.connection.total.maxpoolsize=100

#
# total connections per DBMS for sybase
#
sybase.connection.total.maxpoolsize=100

#
# total connections per DBMS for netcool
#

```

```
netcool.connection.total.maxpoolsize=100

#
# Total connections per DBMS for any other db type that
# is not supported by HP Operation Orchestration
#
custom.connection.total.maxpoolsize=100

#
# Lifetime in seconds for a connection in the pool before
# it gets discarded for oracle.
#
oracle.connection.lifetime=3600

#
# Lifetime in seconds for a connection in the pool before
# it gets discarded for sql server.
#
mssql.connection.lifetime=3600

#
# Lifetime in seconds for a connection for mysql
#
mysql.connection.lifetime=3600

#
# Lifetime in seconds for a connection in the pool before
# it gets discarded for db2
#
db2.connection.lifetime=3600

#
# Lifetime in seconds for a connection in the pool before
# it gets discarded for sybase
#
sybase.connection.lifetime=3600

#
# Lifetime in seconds for a connection to stay in the pool before
# it gets discarded for netcool
#
netcool.connection.lifetime=3600

#
# Lifetime in seconds for a connection in the pool before
# it gets discarded for any other db type that
```

```
# is not supported by HP Operation Orchestration
#
custom.connection.lifetime=3600

# -----
# General configuration per user for DataSource provided
# by a pooled DataSource Provider
# -----

#
# max number of connections a pooled datasource will maintain at a time
#
connection.maxpoolsize=20

#
# min number of connections a pooled datasource will maintain at a time
#
connection.minpoolsize=0

#
# init number of connections when a pooled datasource starts
#
connection.initpoolsize=1

#
# Seconds a Connection can remain pooled but unused before being discarded.
#
connection.maxidletime=300

#
# The number of milliseconds a client to call checkout connection will wait for a
# Connection to be checked-in or acquired when the pool is exhausted
#
connection.checkout.timeout=20000

#
# How many connections at a time connection will try to acquire when
# the pool is exhausted.
#
connection.acquireincrementsize=1

#
# How many times connection pool will try to acquire a new Connection from the
# database before giving up. If this value is less than or equal to zero,
# it keep trying to fetch a Connection indefinitely.
#
```

```

connection.retrycount=1

#
# Seconds connection pool will wait between acquire attempts
#
connection.retrydelay=5

#
# If true, an operation will be performed asynchronously at every
# connection checkout to verify that the connection is valid.
#
connection.validate_oncheckout=true

#
# If true, an operation will be performed asynchronously at every connection
# checkin to verify that the connection is valid.
#
connection.validate_oncheckin=false

#
# If this is a number greater than 0, it will test all idle,
# pooled but unchecked-out connections, every this number of seconds
#
connection.idle_validateperiod=0

#
# If true, a pooled DataSource will declare itself broken and be permanently
# closed if a Connection cannot be obtained from the database after making
# acquireRetryAttempts to acquire one. If false, failure to obtain a Connection
# will cause all Threads waiting for the pool to acquire a Connection to throw
# an Exception, but the DataSource will remain valid, and will attempt to
# acquire again following a call to getConnection().
#
connection.break_afteracquirefailure=true

# #####
# For the flows under Library/Operations/Databases/Microsoft SQL
# The following configuration is for .NET connection pooling.
# #####

#
# number of max connections
#

sqlserver.connection.maxpoolsize=100

```

```
#
# connection lifetime in seconds
#

sqlserver.connection.lifetime=3600
```

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.
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 the Studio:

- Central host server
- Default communication port used
- Choice of HTTP: or HTTPS: (secure sockets) as the Internet protocol

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.

```
dharmarepaircenter.host=localhost
```

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

```
dharmarepaircenter.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.

```
dharmarepaircenter.proto=http
```


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

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.

Backing up OO

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 .jar file.
2. Make a copy of each repository's .jar 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 *Clustering and Load Balancing for HP Operations Orchestration 7.50* (ClusteringGuide.pdf).

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.

Index

Active Directory	
configuring over SSL	3
Administrative tasks	
overview	1
Backup	36
Capabilities	2
Central	
configuring	11
sizing	25
sizing considerations	26
SSO, enabling for	12
Certificate	
Studio, replacing	9
Certificate, security	
replacing	5
Certificates	
testing	10
Clusters	
supporting	36
Concurrency of run schedules	
enabling and disabling	25
Concurrent runs	
configuring maximum allowed	25
maximum, configuring	28
copyright notices	ii
Default ports	2
Groups	2
Heap size	
configuring	29
HP OO	
capabilities	2
groups	2
permissions	2
users	2
LDAP	
configuring over SSL	3
LDAPS protocol	3
legal notices	ii
copyright	ii
restricted rights	ii
trademark	ii
warranty	ii
Log file settings	
configuring	36
Log files	
maximum size, changing	12
Log levels	
configuring	36
Logging levels	
configuring	36
OO	
backing up	36
extended functionality	10
Permissions	2
RAM for Central process	
configuring maximum allowed	25
RAS keystore	
sharing SSL authentication with Central	8
RAS operations	
changing the timeout default	35
Repositories	
switching, enabling	11
restricted rights legend	ii
Scheduler certificate	
replacement	9
Scheduling concurrent runs of a flow	
enabling and disabling	25
Security certificate	
replacing	5
Single sign-on	20, <i>See</i> SSO
using MIT KDC	17
using Windows AD	14
SSL	3
SSL authentication	
sharing between RAS keystore and Central	8
SSO	12, 20
enabling for Central Web application	23
enabling for Central with load balancer	14, 17
starting flows with from outside Central	22
Studio	
reconfiguring	35
Studio certificate	
Studio, replacing	9
Studio.properties file	35
threads	
maximum used by parallel steps, configuring	28

Timeout default		warranty.....	ii
for RAS operations.....	35	Workloads	
trademark notices.....	ii	that include parallel steps, configuring.....	27
Users.....	2		