precisely

EngageOne Server

Installation Guide

Version 4.4 Service Pack 11

Table of Contents

9 - EngageOne Scaling

10 - Enabling SSO to EngageOne applications

11 - Securing network traffic with TLS

12 - Configuring external delivery components

13 - EngageOne Server uninstall

14 - Upgrading EngageOne Server

15 - Troubleshooting Installation

16 - Appendix

1 - Preparing for installation

The following section describes the steps that needed to prepare your server for the EngageOne Server installation.

In this section

Software and hardware requirements

Review the *EngageOne Server 4.4 Software and Hardware Requirements guide* and verify your environment fulfills the hardware and software requirements. If installing on Linux, additional prerequisites are required. Please review the **Special software requirements for Linux** on page 154 section before you begin the installation.

Extracting the release distribution

EngageOne Server is provided in a single .zip file and contains the necessary software components except for the required software mentioned above. Please contact your service representative to obtain the latest release distribution.

The release distribution should be copied and extracted to a temporary location on each application server node you intend to install the software on. The extracted location will be referred to as `<release-distribution>` throughout this document.

The contents of the .zip file are as follows:

```
<release-distribution>/
    /bundles
      files to install each bundle of the system.
    /docs
      various user guides and documentation including this installation guide.
    /install - scripts and configuration files to perform an install of each bundle
        /active-drive
          files to install the system active-drive
        /database
          SQL Server and Oracle .sql scripts to be executed outside of the install scripts
        /groovy
          installation scripts
        /groovy-2.5.6
          Groovy distribution to be installed for executing the installation
        /os-services
          files to install each bundle as an Linux or Windows service at install time
        eos.groovy
          script  to install and configure EngageOne Server
        deploy.properties
          file used at install time to configure the system
        encryptPassword.groovy
          script to encrypt passwords for the deploy.properties file
    /samples
      sample code for developers
    /upgrade
       /database
         database upgrade scripts
       /migration
          migration tools
       /utilities
       /InteractiveEditor
       /KeymapGenerator
```

Pre-install server configuration

Required server ports

EngageOne Server communicates with end users, external components and itself over various network ports. The following ports must be open for the servers with the specified bundle.

Ensure that the appropriate ports are open on all application servers before starting installation of EngageOne bundles.

This charts that follow also specifies the associated configuration properties with which you can modify the specific port number.

See **Establishing System Configuration** for more details.

Incoming and outgoing communications

Port	Protocol	Configuration setting	Bundles	Comments
8080	HTTP	`${core.port}`	all	Incoming and outgoing http communications.
8081	HTTP	`${conversion.port}`	Conversion, Composition, Batch	Incoming and outgoing HTTP communications.
8082	HTTP	`${security.port}`	all	Incoming and outgoing HTTP communications.
8083	HTTP	`${composition.port}`	Composition, Core, Batch	Incoming and outgoing HTTP communications.
1433 / 1521	JDBC	`${db.port}`	Core, Composition	Outgoing database communications. Default port for SQLServer is 1433; Default for Oracle is 1521.
389/636	LDAP / LDAPS	`${ldap.url}`	Core, Security	Outgoing LDAP/LDAPS communications.
25/465/587	SMTP	`${smtp.port}`	Core, Composition, Batch	Outgoing SMTP communications for Email Delivery Channel and administrative alerts.

Port	Protocol	Configuration setting	Bundles	Comments
8080	HTTP	`${designer.services.url}`	Core	Outgoing HTTP communications to the designer services.
6003	TCP/IP	`${vault.port}`	Core	Outgoing communications to the Vault archive system.
<port no>	HTTP	`${eodeliver.services.url}`	Core	Outgoing HTTP communications to the EngageOne Digital Delivery system.
8090	HTTP	`${comparison.docbridge.services.url}`	Core	Outgoing HTTP communications to the DocbridgeDelta comparison utility.

Internal communications between OpenAM components

Port	Protocol	Configuration setting	Bundles
1689	TCP/IP	none	Security
2689	TCP/IP	none	Security
4444	TCP/IP	none	Security
5444	TCP/IP	none	Security
50389	TCP/IP	none	Security
50589	TCP/IP	none	Security
50889	TCP/IP	none	Security
51389	TCP/IP	none	Security
58989	TCP/IP	none	Security
59889	TCP/IP	none	Security

ActiveMQ

Port	Protocol	Configuration setting	Bundles	Comments
8084	HTTP	${notification.port}	Notification	ActiveMQ web console
61617	TCP/IP	none	Notification	ActiveMQ broker port

Linux environment settings

In order to properly handle national characters in file names, (for example, German, or French accents, or Far East characters), please make sure that the file located in:

`/etc/environment` has a environment variable set to appropriate encoding.

For example, `export LC_ALL=en_US.UTF-8`.

Installing scripting prerequisites

The EngageOne Server installation scripts are written using the Groovy scripting language. Groovy is a cross-platform scripting language written on top of the Java Runtime.

Installing Java

Both the install scripts and the applications require a 64-bit Java Runtime Environment (JRE). Please see the *EngageOne Server 4.4 Software and Hardware Requirements* guide for the specific version.

Installing Java in Windows

Perform the following procedure to install Java on a Windows server.

1. Download the installation package from **http://www.oracle.com/technetwork/java/javase/downloads/index.html**.
2. Execute the installation .exe.
3. Set the `JAVA_HOME` environment variable to the Java install location.
4. Add the `JAVA_HOME` environment variable to the existing `Path` variable.

Installing Java in Linux

Perform the following procedure to install Java on a Linux server.

1. Installation of Java differs depending on your version of Linux. Refer to the RedHat or SUSE Linux documentation for installation procedure.
2. Set the `JAVA_HOME` environment variable to Java install location.
3. Add the `JAVA_HOME` environment variable to the existing `Path` variable.

Installing Groovy

Groovy is included in the EngageOne Server release distribution and can be found under `<release-distribution>\install\groovy-<x.y.z>` where `x.y.z` is the required version of groovy.

Installing Groovy in Windows

Perform the following procedure to install Groovy on a Windows server.

1. Copy the Groovy binary distribution from `<release-distribution>\install\groovy-<x.y.z>` to a local file system on the target server.
   For example: `C:\groovy-<x.y.z>`
2. Create a new `GROOVY_HOME` environment variable and set it to the above directory.
3. Add `%GROOVY_HOME%\bin` to your `%PATH%` environment variable.

Groovy is now installed on your Windows server. To verify the Groovy installation was successful, execute the following command in a command prompt:

```
groovy -v
```

This should return something similar to: Groovy Version: x.y.z JVM: 1.8.0_25 Vendor: Oracle Corporation OS: Windows 7

Installing Groovy in Linux

Perform the following procedure to install Groovy on a Linux server.

1. Copy the Groovy binary distribution from `<release-distribution>/install/groovy-<x.y.z>` to a local file system on the target server.
   For example: `/usr/share/groovy-<x.y.z>`
2. Set your `GROOVY_HOME` environment variable to the above directory.
3. Add `$GROOVY_HOME/bin` to your `$PATH` environment variable.
4. Grant execute permissions to Groovy binaries. For example, `chmod -R u+x /usr/share/groovy-2.4.4/bin`.

Groovy is now installed on your Linux server. To verify the Groovy installation was successful, execute the following command in a shell window:

```
groovy -v
```

This should return something similar to: Groovy Version: 2.4.4 JVM: 1.8.0_25 Vendor: Oracle Corporation OS: Linux

Establishing system configuration

Before you begin the installation process, review and complete the necessary configurations in the `<release-medium>\install\deploy.properties` file. This file contains all the configuration settings for a complete EngageOne Server. It should be fully completed before installing any of the nodes that make up the system. Each configuration parameter includes an explanation, example and where appropriate the allowable values.

Property limitations

The `deploy.properties` file is a standard java .properties file with the following limitations:

- Back slashes (\) must be escaped by an additional slash (\\). For example, file system paths in Windows must appear with two slashes: `c:\\Program Files\\EngageOne\\Server\\core`

Understanding install scripts

The following sections will help you to understand the various scripts that will be used during the installation of EngageOne.

The encryptPassword.groovy script

The `encryptPassword.groovy` script is a utility script to encrypt sensitive passwords before placing them in `deploy.properties`. This is an interactive script with no command line options. To encrypt a password simply execute the script and enter your password when prompted. The script will print the encrypted string to a standard output (your shell or console window). Copy and paste this text into `deploy.properties`. A typical execution of this script looks like the following:

```
> groovy encryptPassword.groovy


[INFO ]  Log file location: C:\tmp\logs\encrypt-2016-03-29-170449.log
For security purposes, your password will not be displayed on the screen.
 Enter your password and press <Enter>.
```

```
Password:
Re-enter password:
[INFO ]  Password encryption in progress...
Encrypted password is:
WAbYUKAsmXhwVAyGz1YLM0ArNNZO4LcJ8uviGk6bdg4SoFHyr7whGg==
```

Automatic encryption

In addition to manually encrypting passwords using the ***encryptPassword.groovy*** script described earlier in this section, you also have the option to enter plain passwords directly into the ***deploy.properties*** file. The ***eos.groovy*** script scans ***deploy.properties*** and automatically encrypts plain text values where appropriate, see **The eos.groovy script** on page 14.

> **Note:** Before running ***eos.groovy***, ***deploy.properties*** can contain a mixture of plain and encrypted values. The comment line below is present for each property that can be entered in either encrypted or unencrypted form.
>
> ```
> # An encrypted or plain text value should be supplied.
> ```

In general, all properties with the string ***password*** in the property's name are automatically encrypted, for example, ***eo.db.password***, ***flowable.db.password***, ***ldap.password***, etc. The exception to this, is the optional ***security.encryption.key*** property which is also automatically encrypted.

During execution of ***eos.groovy***, a log with filename format eos***-YYYY-MM-DD-hhmmss.log*** is created, for example, ***eos-2017-12-08-123443.log***. This log contains a section on automatically encrypted plain text values, and will be similar to this:

```
[INFO ] Scanning for plain text passwords in 'deploy.properties' has
started.
[INFO ] Detected 3 new not encrypted password(s).
[INFO ] 3 new encrypted password(s) successfully saved in
'deploy.properties' file.
```

Where all appropriate plain text values have previously been encrypted, the following log entries are issued:

```
[INFO ] Scanning for plain text passwords in 'deploy.properties' has
started.
[INFO ] No new plain text passwords.
```

> **Note:** In certain circumstances, your ***deploy.properties*** file may be used by external programs, for example, when integrating with EngageOne Designer. In this type of scenario, the external program expects passwords to be encrypted and will try to process any plain text password as encrypted strings, causing unpredictable behavior. If your ***deploy.properties*** file is to be used by an external program, it must first be processed by ***eos.groovy*** to ensure that only encrypted passwords are present in the file.

The eos.groovy script

The `eos.groovy` script is used to install and configure EngageOne Server's active-drive and service bundles. It is a target based interface allowing you to perform different operations with different invocations. Below is the usage statement of this script:

```
usage: groovy eos.groovy [options] [target [target2 [target3]...]]
 -b,--bundle <arg>        Specify the name of the bundle to install,
                          validate or configure. Allowed values: [batch,
                          composition, conversion, core, security,
                          notification].
 -h,--help                Usage information.
 -p,--properties <arg>    Specify the location of the deploy.properties
                          file containing configuration information.
 -r,--replicaName <arg>   For clustered environments, the name of the
                          replica node to be installed. Required when type
                          is replica. Ignored otherwise.
 -t,--type <arg>          Type of the node to be configured. Allowed
                          values: [single, primary, replica].
targets:
 active-drive                -> Create and populate the EngageOne active-drive structure.
 configure                   -> Configure the specified bundle. Assumes bundle has already been installed.
 install                     -> Copy the specified bundle's files onto the local machine in preparation for configure.

 validate                    -> Run validation on configuration properties file. Display any errors/warnings.
 downloadWebContainer        -> Download web container to local folder for further local upgrade.
 upgradeWebContainer         -> Upgrade bundle web container version.
 rollbackUpgradeWebContainer -> Rollback the last upgrade of bundle web container.
```

Script Targets

The sections below are a brief overview of each target. For detailed usage, see the **Standalone install** and **Clustered install** sections of this guide.

active-drive

The `active-drive` target creates the shared folder structure. See the **Shared File System (Active Drive)** section for more details.

install

The `install` target copies the necessary software components from the `<release-distribution>` folder to the install folders on the local machine. This target should be run once per bundle, per node.

Installation directories:

- `core.install.dir`
- `security.install.dir`
- `composition.install.dir`
- `conversion.install.dir`
- `batch.install.dir`
- `notification.install.dir`

validate

The `validate` target applies the install script's validation rules without applying them to the installation. This is an optional target and should be used to verify the integrity of

`deploy.properties` and checks various connectivity points to external components from the current server. The validation includes these types of checks:

- Required fields - validation that all required properties are found in the file.
- Data types - validation of URLs, port numbers, and so on, to ensure they are in the correct formats.
- File systems - validation that folders and files specified in the `deploy.properties` file exist and contain the target structures. This includes install folders, active-drive, etc.
- Database connectivity - bundles that require a connection to the database verify the local server can establish a connection.

configure

The `configure` target performs the same validation as `validate` and then applies the configuration to the local server. In the event that one or more property configurations are invalid the script terminates and displays error messages detailing the issues. All invalid configuration is highlighted by a single execution of the script, so that you can fix all errors at one time.

The configure script is intended to be run during the first time install and to apply any subsequent configuration changes. When applying subsequent configuration changes, the script will terminate without making any changes if the bundle service is running. You must stop the service before being able to successfully re-configure the bundle.

Web container upgrade targets

Where any vulnerability is detected on the currently installed bundle's web container Tomcat version, it can be upgraded to the publicly available Tomcat minor version including security patch. This upgrade can be performed via the distributed eos.groovy install scripts by using new targets decribed below.

> **Note:** The Web container upgrade is applicable to all bundles except batch.

upgradeWebContainer

The upgradeWebContainer target backups current bundle's Tomcat version, downloads and installs Tomcat version specified in **upgrade.tomcat.minor.version** property (in deploy.properties file). Internet connection is required for the download to be successful.

downloadWebContainer

In the case when internet download is not available on the server where Tomcat upgrade needs to be performed you can download web container to local folder on other machine and then continue upgrade on the server where bundle is installed by moving downloaded files to server with installed bundle. In order to configure the downloadWebContainer target local folder un-comment **upgrade.tomcat.local.source.path** property (in deploy.properties file) and enter local folder path.

rollbackUpgradeWebContainer

The rollbackUpgradeWebContainer target restores previous Tomcat version from latest backup. Each rollbackUpgradeWebContainer execution will restore prior to currently installed version of Tomcat. Rollback is possible until the version of Tomcat which was included in the release will be restored and no further Tomcat backups are available.

The uninstall.groovy script

The `uninstall.groovy` script is used to uninstall bundles and service bundles. It is a target based interface allowing you to perform different operations with different invocations. Below is the usage statement of this script:

```
usage: groovy uninstall.groovy [options] [target [target2]]]
 -b,--bundle <arg>       Specify the name of the bundle to uninstall.
 -h,--help               Usage information.
 -p,--properties <arg>   Specify the location of the deploy.properties
                         file containing configuration information.
targets:
 bundle          -> Uninstall/delete the specified bundle from the local
 machine.
 service        -> Uninstall the specified service from the local machine
```

Script targets

The sections below are a brief overview of each target. Please see the **Uninstalling EngageOne Server** section for additional usage details.

bundle

The `bundle` target deletes the install folder identified be the `${bundle.install.dir}` property of the specified bundle. All configuration files and temporary data files will be permanently deleted. This target should be run if you would like to permanently remove the install from the current server.

service

The `service` target uninstalls the system service from the machine. In the case of Windows, this uninstalls the Window Service. In the case of Linux, this removes the `/init.d` script and entries.

Log files

In addition to info and error logging to a standard output, all install scripts place a time-stamped log file in the `<release-distribution>\install\logs` folder.

2 - Designer component install and configuration

Before you begin the EngageOne Server installation you must install the Designer Components. For details, see the *Designer Release Notes*.

In this section

Designer Server

EngageOne Server requires the Designer Server to take advantage of the Design Review and Approval feature. Once the Designer Server is installed see the "Designer Services Configuration" section in `deploy.properties`. This URL is used by the Core bundle to retrieve and display Template and Project data.

Comparison services (optional)

You can optionally install and configure a third-party comparison tool to facilitate the Design, Review and Approval feature. You should choose a comparison tool that will generate before and after differences of Designer assets. Refer to **Comparison plug-in support** on page 19 for detailed information on installing and configuring your chosen comparison plug-in.

3 - Comparison plug-in support

Comparison plug-in support is available for use with the Design, Review and Approval feature available from EngageOne Compose or your own custom web application.

You have the choice of using :

- your own custom built comparison plug-in, refer to **Implementing a custom comparison plug-in** on page 20 and for more details on how to write your own implementation refer to the EngageOne Programmer's Reference Guide.
- the supplied comparison plug-ins available in the distribution media, refer to **Implementing sample comparison tools** on page 22

In this section

Implementing a custom comparison plug-in

If required, you can optionally install and configure a third-party comparison tool to facilitate template review when using the Design, Review and Approval feature. The third party comparison tool must be capable of generating before and after differences of Designer Templates.

In order to install and configure your comparison plug-in, you must prepare a java jar file containing the `com.pb.viewpoint.comparison.adapters.ComparisonEngineAdapter` interface custom implementation.

When plug-in is ready, you will typically install the jar file along with its specific transient dependencies on all servers that will run the core bundles. This may occur whilst installing the core bundle or, you may want to add/replace the comparison .jar following the core bundle installation.

The following sections provide information on these activities.

Installing the comparison jar with the core bundle

To install the .jar files whilst installing a bundle: copy the .jar jar file along with its transient jars to the <release-distribution>/bundles\core\plugins\comparison-engine-adapters/<any folder> folder of the unzipped core bundle distribution media. You will then need to install the bundle, refer to **Installing and configuring the Core bundle** on page 55.

Set property in deploy.properties file and run configure core bundle:

```
comparison.adapter.class=
<full name of the
com.pb.viewpoint.comparison.adapters.ComparisonEngineAdapter
implementation class>
```

Add/remove the comparison jar

To add or replace a plug-in after the core bundle installation: copy the .jar files to the:

```
<Bundle installation root folder>
/core/plugins/comparison-engine-adapters/<your chosen folder name>
```

folder of the installed core bundle.

> **Note:** When adding or replacing the plug-in in the core bundle, a restart of the core bundle is required.

Set property in deploy.properties file and run configure core bundle:

```
comparison.adapter.class=
<full name of the
com.pb.viewpoint.comparison.adapters.ComparisonEngineAdapter
implementation class>
```

Implementing sample comparison tools

The set of supplied comparison plug-in implementations are:

- Command line comparison plug-in - pre-installed in the core bundle
- Simple comparison plug-in
- Docbridge Delta plug-in

The plug-in sample implemetations can be found at the following location in the release media:

```
<release-distribution>\samples\comparison-adapters\
```

Configuring the command line comparison plug-in

The command line plug-in allows for the execution of any PDF comparison tool with the following assumptions:

- It can be executed from command line
- Accepts two input PDF files for comparison
- Generates comparison result PDF file.

In order to configure this implementation you need to:

- set the following property in your deploy.properties file and run the core bundle configuration:

```
comparison.adapter.class=
com.pb.viewpoint.comparison.adapters.commandline.CommandLineComparisonEngineAdapter
```

- On all servers that will run the core bundle, enter command profiles to the commandLineAdapter.properties file located in:

```
…/core/plugins/comparison-engine-adapters/command-line-adapter/
```

Profile pattern

In the `commandLineAdapter.properties` file you may enter any number of profiles, which consists of three properties following the pattern shown below:

```
command.line.comparison.adapter.profiles.<anyProfileKey>.name =
<any descriptive name>
command.line.comparison.adapter.profiles.<anyProfileKey>.commandFile =
<full path to command to run>
command.line.comparison.adapter.profiles.<anyProfileKey>.commandParameters =
<command parameters containing placeholders for input files and result file>
        Placeholders'tags:   <ORIGINAL_FILE_PATH_PLACE_HOLDER>,
                             <MODIFIED_FILE_PATH_PLACE_HOLDER>,
                             <RESULT_FILE_PATH_PLACE_HOLDER>
```

Example profiles

The following `commandLineAdapter.properties` file contain examples on how to configure profile commands.

```
command.line.comparison.adapter.profiles.streamDiffDemo.name=StreamDiff from
cdpcom.com  demo
command.line.comparison.adapter.profiles.streamDiffDemo.commandFilePath=
            "C:\\Program Files (x86)\\STREAMdiff\\Bin\\sdConApp.exe"
command.line.comparison.adapter.profiles.streamDiffDemo.commandParameters=
             -sdp C:\\StreamDiffProjects\\PDF.sdp -cd
                    <ORIGINAL_FILE_PATH_PLACE_HOLDER> -td
                    <MODIFIED_FILE_PATH_PLACE_HOLDER> -n -1 -cr -rt PDF -rp
                    <RESULT_FILE_PATH_PLACE_HOLDER>

command.line.comparison.adapter.profiles.DiffPdfc.name=DiffPdfc from
www.qtrac.eu/diffpdfc.html demo
command.line.comparison.adapter.profiles.DiffPdfc.commandFilePath=
            "C:\\Program Files\\diffpdfc\\diffpdfc.exe"
command.line.comparison.adapter.profiles.DiffPdfc.commandParameters=
                    -r <RESULT_FILE_PATH_PLACE_HOLDER>
                       <ORIGINAL_FILE_PATH_PLACE_HOLDER>
                       <MODIFIED_FILE_PATH_PLACE_HOLDER>

command.line.comparison.adapter.profiles.comparepdfcmd.name=
            comparepdfcmd from http://www.qtrac.eu/comparepdfcmd.html demo
command.line.comparison.adapter.profiles.comparepdfcmd.commandFilePath=
            "C:\\Program Files\\comparepdfcmd\\comparepdfcmd.exe"
command.line.comparison.adapter.profiles.comparepdfcmd.commandParameters=
                    -r <RESULT_FILE_PATH_PLACE_HOLDER>
                       <ORIGINAL_FILE_PATH_PLACE_HOLDER>
                       <MODIFIED_FILE_PATH_PLACE_HOLDER>
```

Note: For further information about tools obtained from www.qtrac.eu and cdpcom.com, contact your service representative.

Testing the profile file

The following examples can be used to test if the required configuration settings are correctly set up before actually implementing the plug-in.

```
command.line.comparison.adapter.profiles.justCopyModified.name=
                Copy modified PDF file
command.line.comparison.adapter.profiles.justCopyModified.commandFilePath=copy
command.line.comparison.adapter.profiles.justCopyModified.commandParameters=
                <MODIFIED_FILE_PATH_PLACE_HOLDER>
                <RESULT_FILE_PATH_PLACE_HOLDER>

command.line.comparison.adapter.profiles.justCopyOriginal.name=
                Copy original PDF  file
command.line.comparison.adapter.profiles.justCopyOriginal.commandFilePath=
                copy
command.line.comparison.adapter.profiles.justCopyOriginal.commandParameters=
                <ORIGINAL_FILE_PATH_PLACE_HOLDER>
                <RESULT_FILE_PATH_PLACE_HOLDER>
```

Configuring the simple comparison plugin

The simple comparison implementation is a basic example of pixel-by-pixel page comparison.

To configure the simple comparison plug-in:

- set the following property in your deploy.properties file and run configure core bundle:

```
comparison.adapter.class=com.pb.viewpoint.comparison.adapters.impl.SimpleComparisonEngineAdapter
```

- copy all jar files(including lib folder)

  from:

  `samples/comparison-adapters/simple-adapter/target/`

  to

  `/core/plugins/comparison-engine-adapters/simple-adapter folder`

Configuring the DocBridge Delta plugin

It is important to note that section provides basic configuration instructions for the DocBridge Delta plug-in. You may be required to make additional adjustments to achieve the desired comparison functionality.

To configure the DocBridge Delta plug-in:

- set the following property in your deploy.properties file and run configure core bundle:

```
comparison.adapter.class=
com.pb.viewpoint.comparison.adapters.impl. DocbridgeDeltaAdapter
```

- copy from `samples\comparison-adapters\docbridge-delta`

```
...\target\docbridge-delta.jar
```

```
...\src\main\resources\docBridgeAdapter.properties
```

  to

```
/core/plugins/comparison-engine-adapters/docbridge-delta
```

- set properties in the `docBridgeAdapter.properties` properties file.

4 - Installing the EngageOne Compose Interactive Editor

This section presents an overview of the different editors you can use to add data to an interactive communication and provides detailed information on installing and configuring the Interactive Editor.

In this section

Editing interactive communications

An interactive communication can be edited in the EngageOne Server environment, as follows:

- Using the EngageOne Compose Interactive Editor, a desktop program that is installed on your computer. This provides all of the features of the EngageOne Compose ActiveX embedded editor described below but with the flexibility of browser choice provided by the Cross-Browser option. Refer to **Installing and configuring the Interactive Editor** on page 28.

  Note that the EngageOne Compose Interactive Editor is **not** supported on Microsoft Windows 7 SP1 (32 and 64Bit)

- Using an embedded, fully featured EngageOne Compose ActiveX editor which provides an accurate view of the final printed document. You can make changes to your communication in real-time without the need to open a separate preview of your document. This mode requires the installation of an ActiveX component, which can be installed on demand, but is limited to the use of Internet Explorer. The ActiveX editor is installed automatically and will depend on your implementation's deploy.properties setting. Refer to **deploy.properties setting for interactive editing** on page 39 for details.
- Using the ***Cross-Browser editing*** functionality that supports most mainstream browsers, such as Chrome, Firefox and Edge. In this mode editing is done by selecting options from available prompts, then previewing what the final document will look like. Refer to **deploy.properties setting for interactive editing** on page 39 for details on configuring Cross-browser editing.

The editing experience your users will see when editing communications will depend on the specific properties you configure in your implementation's deploy.properties file, refer to **deploy.properties setting for interactive editing** on page 39.

Points to note:

- Pre 6.6.11 editor versions are not supported. You are advised to uninstall any earlier editor version before performing any installation activity.
- When running the Interactive ActiveX editor for the first time after a 4.4.10 to 4.4.10-P1 upgrade, you may be prompted to perform a machine restart. To ensure the editor is properly upgraded, it is important to perform the restart to avoid software instability. If an immediate restart is not possible, restarting Internet Explorer would suffice.
- **For browsers running under pre-Windows 10 operating systems:** if the Interactive Editor ActiveX control is to be downloaded and installed directly through the browser, or if you wish to use the Interactive Editor desktop program, then Windows update KB2999226 must be installed.
- In a scenario where both the ActiveX and the Interactive Editor are allowed to be used by the organization, to avoid conflicts you must install each editor to separate folders.

Installing and configuring the Interactive Editor

The Interactive Editor can be installed, via:

- The supplied setup program, refer to **The setup program** on page 29.
- Group Policy deployment, refer to **Group Policy deployment** on page 30.
- The command line, refer to **Command line installation and program maintenance** on page 31.

The setup program

Installation

The Interactive Editor setup program can be found in EngageOne Server release material at the following location:

```
.../utilities/Interactive Editor
```

To start the install, double click on the file:

```
setup-engageone-interactive-editor-app.exe
```

The installer provides a dropdown of all supported languages, allowing you to select the installation in your preferred language.

The default install location is:

```
C:\Program Files (x86)\EngageOne\EngageOne Compose\Interactive Editor
App
```

From the **Destination folder** page, you can either leave the default path as it is or, click **Change** to change the install location. Continue following the on-screen instructions to progress and complete the installation.

Upgrading

To upgrade to a new version of the Interactive Editor, double click:

```
setup-engageone-interactive-editor-app.exe
```

You will prompted to confirm the upgrade, click **Yes** to continue the upgrade.

Note: the upgrade will be applied at the current installed location; you will not be prompted for a path change.

Uninstalling

You can uninstall the Interactive Editor using the control panel's add/remove programs option. Alternatively, you will be given the option to remove the program when clicking on the setup file again.

Group Policy deployment

The Interactive Editor can be installed, upgraded or uninstalled on multiple clients using Microsoft's Group policy deployment feature.

It is important to note that the Group Policy deployment requires an msi file rather than an exe file. You will therefore be required to extract the msi from the exe file before using it to perform Group Policy deployment.

There are a number of ways to extract an msi file from an exe file, here we are outlining one possible method.

1. Copy the following file to a local folder:

```
setup-engageone-interactive-editor-app.exe
```

2. Open a command window with administrator access.
3. Navigate to the local folder where the exe file was copied.
4. Execute the following command:

```
"setup-engageone-interactive-editor-app.exe" /s /x
/b"C:\ExtractEditorAppmsi" /v"/qn"
```

   where `c:\ExtractEditorAppMSI` is the file path where the msi will be extracted.

5. Use the resultant msi file as the package to be deployed.

Group Policy distribution of ApplicationStyle.xaml

If you plan to deploy the Interactive Editor using Group policy, then you can also distribute the custom theme file using group policy. For this, you will need to copy your updated `ApplicationStyle.xaml` file to the domain controller server. The source folder of the file will need to be set as a shared drive on the domain controller and the target folder needs to be set as the location on the client's computer, where the program needs to be installed.

Note that you will need to deploy the Interactive Editor on the client's computer before you deploy the theme file.

Command line installation and program maintenance

Installation

To install Interactive Editor from the command line:

1. Copy the following file to a local folder:

```
setup-engageone-interactive-editor-app.exe
```

2. Open a command window with administrator access.
3. Navigate to the folder where the exe file was copied.
4. Execute the following command:

```
setup-engageone-interactive-editor-app.exe /s /v"/qn /norestart /log
install.log INSTALLDIR=\"C:\Program Files (x86)\EngageOne\EngageOne
Compose\EngageOne Interactive Editor App\""
```

   Note that the `INSTALLDIR` parameter is set to the default path where the Interactive Editor will be installed. This path can be changed to a different location in the command where a different install location is required.
5. A new file called "install.log" is created in the same location where the command was run. This file provides logging information about the installation.

Upgrading

If you want to upgrade the Interactive Editor to a new version, simply copy the new exe from the EngageOne Server build and run the command in step 4 above, using administrator access on the newly copied exe. The Interactive Editor will then be upgraded to the latest version.

Uninstalling

Open a command window using administrator access and enter the following command. This will uninstall the Interactive Editor from your computer.

```
wmic product where name="EngageOne Interactive Editor App" call uninstall
```

Configuring the Interactive Editor

The sections that follow describe the steps you need to follow to successfully configure the Interactive Editor detailling mandatory and optional settings.

Refer to **deploy.properties setting for interactive editing** on page 39 and the ***EngageOne Compose configuration Checklist*** document provided in the release distribution for detailed information.

Mandatory configuration settings

The mandatory settings described in this section are found in the deploy.properties file.

- ***Set the editing mode*** `interactive.editor.type=Interactive Editor`

  It is important to note that this setting can also be set to `Mixed` which gives you the option to use either the Interactive or ActiveX editor for editing interactive communications in your operating environment. In this scenario you will also be required to configure the mandatory settings that follow.

- ***Set the title of the editor window.*** The title of the editor window must be defined. It can be localised by using sub-settings. Refer to the ProductTitle setting in **Changing the visual style** on page 37. Refer to the following examples:

```
interactive.editor.application.title=EngageOne Compose interactive editor
interactive.editor.title.fr=Application de l'éditeur interactif
```

- ***Set the minimum version of the Interactive Editor*** as follows:

```
interactive.editor.version=6.6.8
```

- ***Set the address of the web socket the browser is bound to***. Either WebSocket (ws) or WebSocked Secure (wss) resource identifier can be used for unencrypted or encrypted connections. Refer to the following examples:

```
interactive.editor.socket.address=ws://localhost.pbeo.net:8080/relay-service/cie
interactive.editor.socket.address=wss://localhost.pbeo.net:8080/relay-service/cie
```

- ***Set the connection time-out and the response time-out settings***. Refer to the following examples.

```
interactive.editor.connection.timeout=5
interactive.editor.response.timeout=10
```

- ***Download the editor setup***

  This setting determines whether the Interactive Editor setup program can be downloaded directly from the webpage. By default, this feature is disabled. Enable this setting if users have sufficient privileges to install the application.

  Refer to the following example:

```
interactive.editor.show.download.link=false
```

Optional setting

The optional setting described in this section are found in the `deploy.properties` file.

Setting additional cookies

You can optionally define the names of additional cookies used by the Interactive Editor to access resources from EngageOne Server. It must either have a value or it must be commented out, i.e. if a parameter exists, it cannot be empty.

Refer to the following example:

```
interactive.editor.cookie.names=cookie1,cookie2
```

Logging and caching

The Interactive Editor uses a configuration file called `log4net.config`. This file is used to configure the logfile path and the log level. This configuration file is created and stored in the installation folder of Interactive Editor, for example:

```
C:\Program Files (x86)\EngageOne\EngageOne Compose\EngageOne Interactive
 Editor App\log4net.config
```

This log file will be retained on upgrade so that any previous configuration changes you have made are preserved.

Logfile Path

The path of the log file is set in the `<file>` element within `<appender>`, default settings shown below:

Example config file

```
<log4net>
  <root>
    <level value="INFO" />
    <appender-ref ref="console" />
    <appender-ref ref="file" />
  </root>
  <appender name="file" type="log4net.Appender.RollingFileAppender">
    <file type="log4net.Util.PatternString"
                value="%property{DefaultLogPath}\session.log" />
    <appendToFile value="true" />
    <rollingStyle value="Size" />
    <maxSizeRollBackups value="5" />
    <maximumFileSize value="10MB" />
    <staticLogFileName value="false" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date %level %logger - %message%newline" />
    </layout>
  </appender>
</log4net>
```

The `<file> value` attribute may be changed as required, however, you must bear in mind that when changed you must ensure that the path given is accessible to the user of the Interactive Editor.

The `{DefaultLogPath}` property is resolved by the Interactive Editor to:

```
%temp%\IEA Cache\sessions\<datetime>
```

Log level

The level of logging is controlled by the `<level> value` attribute, which can be one of the following:

- ALL - Every level of message will appear in the log
- DEBUG - Debug messages and all those below will be logged. Very verbose.
- INFO - No debug, informational messages and all those below will be logged. The shipped default.

- WARN - No informational or debug messages. Warning messages and all those below will be logged.
- ERROR - Only error or fatal messages will be logged.
- FATAL - Only fatal level messages
- OFF - No logging at all, although there will still be a blank file created

Caching

The Interactive Editor's cache will be stored in:

```
C:\Users\<WindowsUserId>\AppData\Local\Temp\7\IEA Cache\
```

By default Interactive Editor's log files and cached resources will be stored in a unique session folder, for example:

```
C:\Users\<WindowsUserId>\AppData\Local\Temp\7\IEA
Cache\sessions\20180823095854040\session.log
C:\Users\<WindowsUserId>\AppData\Local\Temp\7\IEA
Cache\sessions\20180823095854040\ie_cache\ipe.resource.493675A18BBA401AAE4D569BDE9BFFE2
```

Changing the visual style

Part of the appearance of the Interactive Editor Application is determined by settings in a file called `ApplicationStyle.xaml`. This file is located in the installation folder at:

```
Themes\InteractiveEditor
```

By default the settings represent the Precisely theme. If you are writing your own web application, you may want the Interactive Editor Application to more closely represent the appearance of your web page.

Note that you can deploy `ApplicationStyle.xaml` using Group Policy deployment, refer to **Group Policy distribution of ApplicationStyle.xaml** on page 30

The table below outlines the available settings:

Setting	Description
ProductFamily	This text is displayed in the loading screen.
ProductName	This text is displayed in the loading screen below the ProductFamily text.
ProductTitle	This text can be displayed in the title bar along with the server name and the name of the template being edited
WindowTitleFormat	Defines the format of the text to be displayed in the title bar. The {ProductTitle} field will be replaced with the ProductTitle setting from this file. The {Server}, {Template} and {WindowTitle} fields will be replaced with the appropriate parameters supplied to the editor when launched.
CaptionBackgroundBrush	The color of the title bar and the loading screen background. The default value uses a gradient effect. If you wish to use a single color, change this to a SolidColorBrush as shown in the example below. You can use the #rrggbb notation to specify the color, though well-known color names can also be specified literally.

Example ***CaptionBackgroundBrush*** usage:

```
<SolidColorBrush x:Key="CaptionBackgroundBrush" Color="#157fbd"/>
```

CaptionForegroundBrush	The color of items that appear on areas in the CaptionBackgroundBrush color, for example, the title bar text and the minimize and maximize buttons.

Setting	Description
InactiveCaptionBackgroundBrush, InactiveCaptionForegroundBrush	The title bar colors to use when the application is deactivated. If you have changed the CaptionBackgroundBrush then you will need to change the InactiveCaptionBackgroundBrush and maybe also the InactiveCaptionForegroundBrush, as shown in the example that follows.

Example ***InactiveCaptionBackgroundBrush, InactiveCaptionForegroundBrush*** usage:

```
<SolidColorBrush x:Key="InactiveCaptionBackgroundBrush" Color="#9dbddb"/>
<SolidColorBrush x:Key="InactiveCaptionForegroundBrush" Color="#f0f0f0"/>
```

ContainerBackgroundBrush, ContainerForegroundBrush	The color of the background and text in the ribbon bar and the status bar.
SelectedItemBrush, MouseOverBrush, MousePressedBrush, LineBrush, FaintLineBrush	The colors used to render the controls in the ribbon bar.

Changing the application icon

If you wish to change the application icon, you can modify the supplied Application.ico file in:

```
Themes\InteractiveEditor
```

When changing the icon, be aware that Windows Explorer caches application icons so if you have run the application with the default icon you may need to logout out of Windows and back in again to see the change.

deploy.properties setting for interactive editing

deploy.properties setting	Description
interactive.editor.type	Important note: Previously (versions below 4.4.8) the `activex.enabled` setting was used instead of this setting. This is now obsolete and cannot be used Possible values: `Interactive Editor` - Interactive Editor installed under Windows. `ActiveX` - ActiveX Editor where possible (Internet Explorer) otherwise uses the cross browser editor `Mixed` - uses the Interactive Editor and optionally ActiveX Editor `None` - use cross browser editor only.
interactive.editor.socket.address	Determines the address the web socket the browser is bound to. This property is required and used only for `Interactive Editor` and `Mixed` types.
interactive.editor.title	Sets the Interactive Editor window title. It can be localised by using sub-settings with language code appended to the property name. This property is required and is used only for the `Interactive Editor` and `Mixed` types.
interactive.editor.version	Sets the required version of the Interactive Editor. This property is required and used only for the `Interactive Editor` and `Mixed` types.
interactive.editor.connection.timeout	Specifies the amount of time (in seconds) the web page will wait for the web socket service response. This property is required and used only for `Interactive Editor` and `Mixed` types.
interactive.editor.response.timeout	Specifies the amount of time (in seconds) the web page will wait for the response from Interactive Editor. This property is required and used only for `Interactive Editor` and `Mixed` types.
interactive.editor.cookie.names	Specifies via a comma-separated list of names of additional cookies which will be used by Interactive Editor to access resources from EngageOne server. This property is used only for `Interactive Editor` and `Mixed` types and is optional.

deploy.properties setting	Description
interactive.editor.show.download.link	Specifies if the user can download the Interactive Editor setup program directly from the webpage. Enable this setting if the users have sufficient privileges to install the application. This property is required and used only fof `Interactive Editor` and `Mixed` types.
interactive.editor.pulse	This property specifies the interval (in seconds) the IEA sends a connection sustain signal to the Relay-Service. This property is used only for Interactive Editor and Mixed types and is optional. Use this parameter when, the connection between the Interactive Editor Application and Relay-Service is broken after several minutes of inactivity in the Editor application. **Recommendation:** set this parameter to half the interval of inactivity which triggered the break in connection.

5 - External component install and configuration

In this section

Load balancer

EngageOne Server's architecture is designed with scalability and failover in mind. To take advantage of clustering you must install and configure your own hardware or software load balancer. The following configurations are required:

- Sticky Sessions - all bundles require that cookie type sticky sessions are enabled. In the event of a node going offline, if the load balancer is configured correctly the user can failover gracefully to another node. Sticky sessions are required for improved performance and to ensure a consistent user experience.
- Idle Connection Timeout- if your load balancer supports an idle connection timeout, ensure that the timeout value is large enough to support lengthy EngageOne operations. An idle connection timeout of 300 seconds is recommended.

EngageOne Server provides a simple status and health check endpoint you may use to perform availability checks from the load balancer. For more information on this endpoint, please see the **Server Status and Health Check Endpoint** appendix.

Note: A Load balancer is only used in an EngageOne clustered installation.

Configuring retries

To improve the quality of High Availability Cluster installation it is reommended to configure the retry mechanism on load balancers.

Where a node within the bundle is no longer active and the retry mechanism has not been configured, the load balancer will still see the node as being active. On any subsequent call to this node, the load balancer will return an error indicating that the node is not active (e.g HTTP 502), this error will be passed on to the user. To avoid this scenario the load balancer must be configured to use the retry mechanism which will result in the load balancer calling an alternative active node to ensure high availability is maintained.

Setting retry on the F5 load balancer for all pools related to EnageOne in F5 admin console execute:

Local Traffic -> **Pools** -> **Pool List** -> **[specified pool]** -> **Configuration Advanced** -> **Reselect Tries**

Set **Reselect Tries** to the number equal number of members in specific Pool minus 1. For example, if the security bundle consists of three members (3 security bundle instances), then the **Reselect Tries** value should be set to 2.

LDAP Server

EngageOne requires an LDAP v3 directory server to connect to for authentication. Active Directory is primarily supported, though support extends to LDAP v3 directory servers in general. By default, EngageOne expects the schema to follow these rules:

1. The `cn` attribute exists and contains values that are used as display names.
2. The `sAMAccountName` attribute exists and contains `username` values.
3. The `member` attribute exists and contains member values.
4. The `memberOf` attribute exists and contains membership values.
5. The `mail` attribute exists.

Installation and configuration against an LDAP v3 directory server using a different attribute schema is possible, refer to **Using an alternative attribute schema** on page 43. Currently, EngageOne supports only a single instance LDAP. There is no additional configuration needed beyond an LDAP that meets these requirements and is network accessible by the EngageOne and OpenAM installations.

LDAP groups visible in OpenAM should contain less than 1,000 members.

> **Note:** Use the `ldap.groupFilter` property in `deploy.properties` to define which LDAP group is exposed to EngageOne.

Using an alternative attribute schema

If you wish to install EngageOne using a different attribute schema, custom settings are required for both EngageOne and OpenAM. This section outlines the steps you will need to follow.

LDAP configuration in EngageOne

The following deploy.properties configuration provides guidelines on the settings that need to be configured. You may need to make certain adjustments based on your specific operating requirements.

```
# Directory user filter
# The search filter to append to the default filter when loading users.
 The
# default search filter is created using the attribute specified by
# ldap.usernameField. For example, if the username field is "uid", then
# the default search filter would be "(uid={0})"
# This property only needs to be changed if your Directory does not use
# the 'Person' objectCategory for users.
# Default:
```

```
#   ldap.userFilter=(&(objectCategory=Person))
ldap.userFilter=(&(objectClass=person))

# Directory group filter
# The filter to append to the default filter when loading groups. The
# default group search filter is created # using the attribute specified
# by ldap.groupNameField. For example, if the group name field is "cn",
 then
# the default group search filter would be "(cn={0})" where {0} is
dynamically
# replaced with the group name being searched for.
# This property only needs to be changed if your Directory does not use
# the 'Group' objectCategory for groups.
# Default:
#   ldap.groupFilter=(&(objectCategory=Group))
ldap.groupFilter=(&(objectClass=groupOfNames))

# Directory name attribute
# The name of the directory field that holds the user's name.
# This property only needs to be changed if your Directory does not use
# the 'cn' attribute for users.
# Default:
#   ldap.displayName.attribute=cn
ldap.displayName.attribute=cn

# Directory account name attribute
# The name of the directory field that holds the user's account name.
# This property only needs to be changed if your Directory does not use
# the 'sAMAccountName' attribute for user account names.
# Default:
#   ldap.username.attribute=sAMAccountName
ldap.username.attribute=uid

# Directory group member attribute
# The name of the directory field that holds the members of a group.
# This property only needs to be changed if your Directory does not use
# the 'member' attribute for group members.
# Default:
#   ldap.member.attribute=member
ldap.member.attribute=member

# Directory e-mail address attribute
# The name of the directory field that holds the user's email address.
# This property only needs to be changed if your Directory does not use
# the 'mail' attribute for user e-mail addresses.
# Default:
#   ldap.email.attribute=mail
ldap.email.attribute=mail
```

OpenAM configuration

1. In EngageOne datastore change attributes **LDAP Users Search Attribute** and **Authentication Naming Attribute** from **sAMAccountName** to value defined in `ldap.username.attribute` property. Refer to the `ldap.username.attribute` entry in **LDAP configuration in EngageOne** on page 43.
2. Set as empty **Attribute Name for Group Membership** and **Attribute Name of Group Member URL**.
3. Set value defined in `ldap.member.attribute` to **Attribute Name of Unique Member** and **Default Group Member's User DN**. Refer to the `ldap.member.attribute` entry in **LDAP configuration in EngageOne** on page 43.

Agent Configuration

*In **OpenAM EngageOneAgent/ Application/ Profile Attributes Processing / Profile Attribute Mapping** remove the following:*

```
[sAMAccountName]=viewpoint_userId
```

add:

```
[<ldap.username.attribute>]=viewpoint_userId
```

Here, we are adding the `ldap.username.attribute` which we previously defined in the deploy.properties file. Refer to **LDAP configuration in EngageOne** on page 43.

Shared File System (Active Drive)

Create Shared Folder (Clustered installation only)

On a machine dedicated for active-drive store, create a network share, such as `\\server\share\active-drive`. Grant permissions to this share to ensure that the user performing the install and the user running the bundle services can read and write files beneath the `${active.drive.dir}` location. The `${os.service.username}` and `${os.service.password}` properties in the install configuration file allow you to configure a specific user with which to run the bundle services. If these properties are commented out, the bundle services will run as the root user on Linux and as the LocalSystem account on Windows.

Install Active Drive

1. Verify the `${active.drive.dir}` value in `deploy.properties` is populated correctly. For a clustered environment, this should be a shared location. For a standalone install, this can be a folder on the local file system.

Attention: When installing on Windows, you must use a UNC path for `${active.drive.dir}` rather than a mapped drive to access the shared file system. Drive mappings are only restored on Windows logon, which the service user does not do.

2. From the `<release-distribution>\install` folder execute the following command:

```
groovy eos.groovy -p <path to deploy.properties> active-drive
```

A successful execution will output the following:

```
[INFO ]  Log file location:
C:\tmp\install\logs\active-drive-2016-03-29-172847.log
[INFO ]  Validating properties
[INFO ]  Validating properties for
com.pb.deploy.bundle.validator.rules.ActiveDriveCreateValidationRules

     [copy] Copying 4 files to C:\tmp\active-drive
     [copy] Copied 11 empty directories to 8 empty directories under
C:\active-drive
[INFO ]  Active drive has been created successfully at
\\shared-file-system\active-drive
```

3. Verify the newly created active-drive has a top-level structure similar to:

```
${active.drive.dir}/
  /config
  /conversionJobs
  /dictionary
  /tds
  /temp
  /promotion/import
  /promotion/export
```

Databases

EngageOne Server requires two databases. The "eos" database is the primary database storing administrative, interactive and Template content data. The "flowable" database backs a third party BPMN engine for Interactive Workflow and Design Review and Approval functionality.

Creating database users

Create a database user with which the application software should access the database.

Microsoft SQL Server

Give the database user membership of the following database roles on both the "eos" and "flowable" databases:

- `db_datareader`
- `db_datawriter`
- `db_ddladmin`

In addition, the database user must have permission to execute all stored procedures in the "eos" database. The database administrator should grant this permission using their preferred method from the following options:

- Give the database user membership of the db_owner role, or
- Grant EXECUTE permission on the dbo schema to the database user, or
- Grant EXECUTE permission on each stored procedure to the database user.

Oracle

Create two Oracle users/schemas (eos and flowable) for use by the application. The names should not contain special characters. Grant the following permissions to these users:

- Create and alter object privileges within the schema (such as CREATE TABLE).
- Insert, read, update and delete data privileges on the tables within the schema.
- Execute privileges on the stored procedures, functions and packages within the schema.

> **Note:** When connecting EngageOne Server to the Oracle databases it is required that the runtime user has access to only its respective schema (eos or flowable) that is used for the EngageOne instance. You should not associate that database user to a role that would violate this requirement. For example, the DBA role must not be kept as a role on the database user.

Attention: Make note of the user-name and password as these will be populated in the `${eo.db.username}`, `${eo.db.password}`, `${flowable.db.username}`, and `${flowable.db.password}` properties of `deploy.properties`.

Create and populate the eos database

Copy database scripts from `<release-distribution>\install\database\<mssql or oracle>` scripts to a temporary location on the target database server.

Creating an eos database in Microsoft SQL Server

The following steps must be performed to properly setup the database that will be used by EngageOne Server. It is anticipated that these steps will be performed by your database administrator. The user executing the database installation script should have the following capabilities:

1. As a database system administrator create an empty database. For example: eos.
2. Execute `EngageOne-mssql-db-install.sql` against the newly created database. For more information on running SQL database scripts, see your Microsoft Server documentation.
3. Successful script execution will result in the database being populated with tables, indexes and procedures.

Creating an eos database in Oracle

The following steps must be performed to properly setup the database that will be used by EngageOne Server.

1. Create a new connection for your user-name and execute `EngageOne-oracle-db-install.sql` against the newly created schema.
2. Successful script execution will result in the database being populated with tables, indexes and procedures.

Oracle database schema validation

Once execution of database installation or update script has completed, the following SQL command should be executed:

```
SELECT * FROM DBA_OBJECTS WHERE OWNER = '<DB_schema_owner>' AND STATUS
= 'INVALID';
```

Before the command is executed, ensure that the `<DB_schema_owner>` item above is replaced with a proper owner name – this is the owner of your EngageOne database schema (your login to EngageOne Oracle database). If the above command returns any rows, it is recommended to recompile the entire database schema by executing the following command:

```
EXEC DBMS_UTILITY.compile_schema(schema => '<DB_schema_owner>');
```

Ensure that the `<DB_schema_owner>` item above is replaced with a proper owner name. When the compilation has finished, the first SQL command described above must be run once again:

```
SELECT * FROM DBA_OBJECTS WHERE OWNER = '<DB_schema_owner>' AND STATUS
= 'INVALID';
```

Ensure that the `<DB_schema_owner>` item above is replaced with a proper owner name. You may proceed with further installation steps if no rows are returned. You will be required to investigate further is rows are returned.

Creating an flowable database

The following steps must be performed to properly setup the flowable database.

Microsoft SqlServer

- As a database system administrator create an empty database. For example: flowable.

It is recommended to set transaction isolation level for flowable database to `READ_COMMITTED_SNAPSHOT` to avoid deadlock issues.

```
ALTER DATABASE flowable SET READ_COMMITTED_SNAPSHOT ON
```

Oracle

- Database schema is defined by username. No further action is required.

Attention: Make note of the databases created as these will be populated in the `${eo.db.name}` and `${flowable.db.name}` properties of `deploy.properties`. For oracle `${eo.db.name}` and `${flowable.db.name}` is the SID (Oracle System ID) from the connection info.

Configure connection thresholds

Below are the default minimum and maximum database pool configurations of each bundle. These settings are tuned to 250 users per node. Verify your SQL Server or Oracle is configured to allow and handle these connections. If desired, see "Tuning the System for Performance" in the *EngageOne Server Administration Guide* to tune these defaults.

Table 1:

Bundle	eos min	eos max	flowable min	flowable max
Security	N/A	N/A	N/A	N/A
Core	10	200	10	200
Composition	10	513	N/A	n/a
Conversion	N/A	N/A	N/A	N/A
Batch	0	20	N/A	N/A
Notification	N/A	N/A	N/A	N/A

Note: N/A indicates the bundle does not connect to the associated database.

Other database considerations

Where EngageOne is installed in a MS SQL Server environment, you can use integrated security. In such a scenario the deploy.properties file should be configured with the following properties settings:

```
db.connection.type=integratedSecurity
```

You will also need to set the following properties accordingly:

- `os.service.username`
- `os.service.password`

Note that when integrated security is used the following properties should be removed or commented out:

- `eo.db.username`
- `eo.db.password`
- `flowable.db.username`
- `flowable.db.password`

You can find more details in comments in deploy.properties file. If Integrated security is not used it is necessary for the SQL Server to be configured in SQL Mixed Authentication Mode. If you use a firewall, you must ensure that the database server ports are open.

6 - EngageOne standalone install

A standalone or single server installation is suited for demonstration, development and test type scenarios. A clustered installation is recommended for production environments.

In this section

Pre-installation steps

Note: Windows Server 2016 - EngageOne Server requires access to Active Drive which is a network shared directory. The default local user accounts do not have access to network resources on Windows Server 2016. It is therefore important to ensure that a access to network resources is available for users running EnageOne services.

Before you begin be sure you have executed the following steps.

1. Populated `deploy.properties` with the necessary configurations related to your particular operating requirements.

   Note: It is mandatory that the deploy.properties setting `security.valid.goto.resources` is configured. This property must contain at least one valid URL, which is used for OpenAM redirects. Note that in clustered environment, you can add multiple commas separated valid URLs, eg.

   ```
   http://.another.net://,http://.eo.local://
   ```

2. Installed Java and Groovy on the target application server.
3. Created the Active Drive and EngageOne databases.

Installing and configuring bundles

The installation process for each bundle is relatively similar. The following describes the general steps, followed by specific details for each bundle.

Note: ***Installing in a Windows environment -*** you must run the installation scripts as a user in the Administrators group.

Installing in a Linux environment - depending on your operating requirements, you can install EngageOne Server either as a root or non-root user. If you choose to install as a non-root user, you must ensure that appropriate permissions are assigned. Refer to **Running the installation as a non-root privileged Linux user** on page 157 for detailed information.

Install bundle

- First, execute the `install` target to copy the specified bundle to the target install location (for example, `${security.install.dir}`).

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> install
```

> **Note:** If the bundle is already installed, the install target will delete the binaries and temporary files before installing. Bundle configuration and data remain intact.

Validate configuration (Optional)

- (Optional) run the `validate` target to check the configuration properties you have provided. See the description of **validate** for more information.

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> validate
```

> **Note:** The configure step performs these validations and prevents you from proceeding if any configuration properties have not been specified correctly.

Configure Bundle

The final step in the install process is to run the `configure` target. For a standalone installation, provide the node type (`-t`) value `single`. This step performs the following:

- Configures the bundle in the target install directory from the values provided in `deploy.properties`.
- Creates a Windows or Linux service to start and stop the application server (excluding the Batch bundle).

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> -t single
configure
```

Start the service

The `configure` target will install the bundle as a System Service (Windows Service on Windows or SysVinit script on Linux). For details on starting and stopping services, see the **System Services** appendix.

Installing and configuring the Security bundle

The Security bundle is a prerequisite to all other bundles and should be installed first. Unlike the Core, Conversion and Composition bundles, the installation requires this service is running to properly configure the underlying OpenAM Server. The service is started by the script, and may take a few minutes to initialize.

1. Install the Security bundle by executing the following command:

```
groovy eos.groovy -b security -p <deploy.properties path> install
```

2. (Optional) validate the installation of the Security bundle by executing the following:

```
groovy eos.groovy -b security -p <deploy.properties path> validate
```

3. Configure the Security bundle by performing the following:

```
groovy eos.groovy -b security -p <deploy.properties path> -t single
configure
```

> **Note:** If configuration of the Security bundle does not complete, see **Script configuring the Security bundle never completes** on page 143 for troubleshooting steps.

Validate the security installation was successful by performing the following:

4. Check the system status by opening your browser and navigating to the status application - **http://<server>:<port>/status**
   The page should load and display the system verifications. For additional details, see the **Server Status and Health Check Endpoint** appendix.
5. Check OpenAM EngageOne realm.
   a) Open your browser and navigate to the OpenAM administration console - **http://<server>:<port>/OpenAM**
   b) Login using the user `amadmin` and `password` `${security.admin.password}` from `deploy.properties`.
   c) Click the **EngageOne** realm.
   d) Click **Subjects** in the left hand pane.
   e) Verify you see a list of users and groups from your LDAP server.

Changing the default LDAP container values

If you are using a different LDAP container than *users* (the default), two settings must be changed.

1. Navigate to `UnzippedInstallDir\install\groovy\lib\resources`

   Open `openam-config.properties`

2. Remove the `users` value from the following:

   - Change `sun-idrepo-ldapv3-config-people-container-value=users`

     to `sun-idrepo-ldapv3-config-people-container-value=`

   - Change `sun-idrepo-ldapv3-config-group-container-value=users`

to `sun-idrepo-ldapv3-config-group-container-value=`

Installing and configuring the Core bundle

The Core bundle component provides the primary interfaces to the EngageOne Server features.

1. Install the Core bundle by executing the following:

```
groovy eos.groovy -b core -p <deploy.properties path> install
```

2. (Optional) validate the installation of the Core bundle by executing the following:

```
groovy eos.groovy -b core -p <deploy.properties path> validate
```

3. Configure the Core bundle by performing the following:

```
groovy eos.groovy -b core -p <deploy.properties path> -t single
configure
```

4. Start the service.
   For additional information on starting the service, see the **System Services** appendix.

Validate the Core installation was successful by performing the following:

5. Check the system status by opening your browser and navigating to the status application - **http://<server>:<port>/status**
   The page should load and display the system verifications. For additional details, see the **Server Status and Health Check Endpoint** appendix.
6. Check the Interactive login.
   a) Open your browser and navigate to the EngageOne Interactive application - **http://<server>:<port>**
   b) Login using `${initial.sysadmin.username}` and the corresponding password.
   c) Click **Communities**.
   d) Click **Create** to create a new community.

      Note: The Community administrator may be a user of your choosing.

7. Check EngageOne Administration access.
   a) Open your browser and navigate to the EngageOne Interactive application - **http://<server>:<port>**
   b) Login using `${initial.sysadmin.username}` and the corresponding password.

c) Click **EngageOne Admin**.
Verify a new tab opens and you can access the EngageOne Administration application.

Installing and configuring the Composition bundle

The Composition bundle is the primary rendering and delivery engine for EngageOne Interactive.

1. Install the Composition bundle by executing the following:

```
groovy eos.groovy -b composition -p <deploy.properties path> install
```

2. (Optional) validate the installation of the Composition bundle by executing the following:

```
groovy eos.groovy -b composition -p <deploy.properties path> validate
```

3. Configure the Composition bundle by performing the following:

```
groovy eos.groovy -b composition -p <deploy.properties path> -t single
 configure
```

4. Start the service.
For additional information on starting the service, see the **System Services** appendix.

Validate that the Composition installation was successful by performing the following steps:

5. Check the system status by opening your browser and navigating to the status application - **http://<server>:<port>/status**.
The page should load and display the system verifications. For additional details, see the **Server Status and Health Check Endpoint** appendix.

Installing and configuring the Conversion bundle

The Conversion bundle supports the attachments feature in the out-of-the-box Interactive application.

Note: Before installing the Conversion bundle on Linux, ensure you have the libpng requirement installed as described in the **Software Requirements** section.

Note: Non-embedded fonts used in PDF attachments need to be available on the Linux system. Install a TrueType "Windows fonts" package on your Linux system. For example, http://corefonts.sourceforge.net/.

1. Install the Conversion bundle by executing the following:

```
groovy eos.groovy -b conversion -p <deploy.properties path> install
```

2. (Optional) validate the installation of the Conversion bundle by executing the following:

```
groovy eos.groovy -b conversion -p <deploy.properties path> validate
```

3. Configure the Conversion bundle by performing the following:

```
groovy eos.groovy -b conversion -p <deploy.properties path> -t single
 configure
```

4. Start the service. For additional information on starting the service, see the **System Services** appendix.

Validate that the Composition installation was successful by performing the following:

5. Check the system status by opening your browser and navigating to the status application - **http://<server>:<port>/status**
   The page should load and display the system verifications. For additional details, see the **Server Status and Health Check Endpoint** appendix.

Installing and configuring the Batch bundle

The Batch bundle is a series of command line utilities to execute bulk document generation and perform system cleanups. There is no associated service. The Batch bundle includes the EngageOne Server Batch and Purge programs.

1. Install the Batch bundle by executing the following:

```
groovy eos.groovy -b batch -p <deploy.properties path> install
```

2. (Optional) validate the installation of the Batch bundle by executing the following:

```
groovy eos.groovy -b batch -p <deploy.properties path> validate
```

3. Configure the Batch bundle by performing the following:

```
groovy eos.groovy -b batch -p <deploy.properties path> -t single
configure
```

4. The Batch bundle is a series of command line utilities to execute bulk document generation and perform system cleanups. There is no associated service.

Installing and configuring the Notification bundle

The Notification Bundle provides the message queue infrastructure that enables notification messages to be distributed to external systems.

1. Install the Notification bundle by executing the following command:

```
groovy eos.groovy -b notification -p <deploy.properties path> install
```

2. (Optional) Validate the installation of the Notification bundle by executing the following command:

```
groovy eos.groovy -b notification -p <deploy.properties path> validate
```

3. Configure the Notification bundle by performing the following command:

```
groovy eos.groovy -b notification -p <deploy.properties path> -t single
 configure
```

4. Start the service.
   For additional information on starting the service, see **System Services**.

   Validate the Notification installation was successful by performing the following:

5. Check system status by opening your browser and navigating to the status application:
   http://<server>:<port>/status

   The page should load and display the system verifications.

6. Verify that ActiveMQ message broker is running by browsing to the ActiveMQ application:
   http://<server>:<port>/activemq

   The EngageOne logon page should be displayed.

   Log on with suitable credentials. The ActiveMQ web console should then be displayed.

By default, notification messages are sent to the ActiveMQ instance in the Notification bundle. It is possible to configure the system to use an external JMS-compliant message broker instead of the default ActiveMQ instance. If an external JMS broker is used, the Notification bundle is not required and it should not be installed.

The following steps should be carried out before installing the Core and Batch bundles so that they can publish messages to the external message broker.

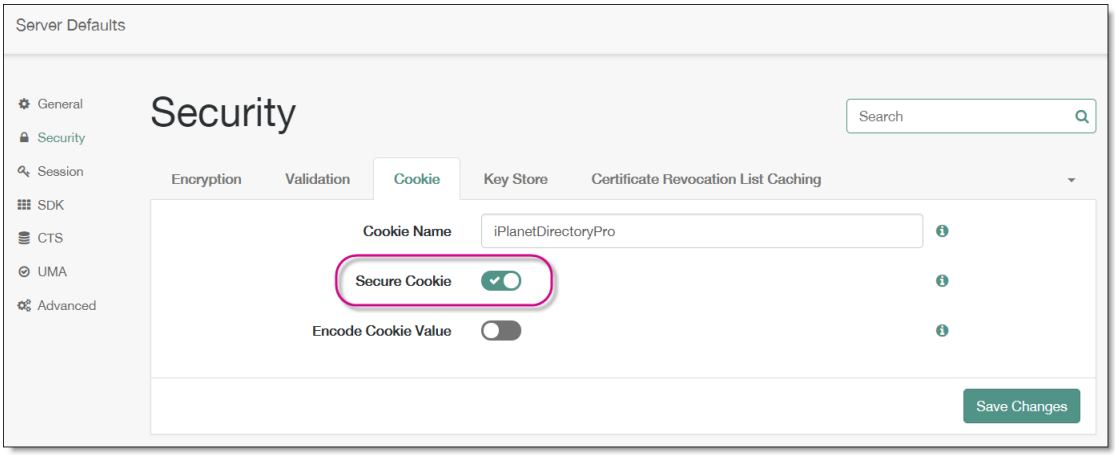
- Create a JAR file containing a class that implements the JMS `ConnectionFactory` interface. See the *Programmer's Reference Guide* for information about creating this class.
- After unzipping the installation media, copy the JAR file to the `/plugins/notification-connection-factory` folder in both the Core and Batch bundle's installation folder. The JAR file will then be copied to the correct target location for each bundle when the bundle is installed. Note that the JAR must be installed on all servers where the Core or Batch bundles will be installed.
- Update the following properties in deploy.properties before installing any bundles:
  - `notification.connectionFactory.class` – set this to the name of the connection factory class within the custom JAR.
  - `notification.jms.broker.url` – set this to be the URL of the external message.

Securing cookies in OpenAM

Authentication may be compromised if the secure flag is not set.

You can configure the OpenAM server to use secure cookies by following these steps:

1. Navigate to the OpenAM console.
2. From the top toolbar, click **Configure** > **Server Defaults**.
   The **Server Defaults** page is displayed with the **General** options open.
3. Click on **Security** to open the security options.
4. Click the **Cookies** tab.
5. Click the **Secure Cookie** toggle to set secure cookies.



6. Click **Save Changes**.

7 - EngageOne clustered install

EngageOne Server clustered architecture is a primary replica configuration. A single primary node for each bundle must exist, with one or more additional replica nodes.

Attention: It is recommended that you review the steps and if possible perform a standalone install before proceeding to a full clustered install. The following steps assume a basic knowledge of the different bundles and installation steps.

In this section

Cluster specific configuration properties

Before you begin your clustered install, review and complete the "Service Communication Configuration" and "Clustering Configurations" sections of `deploy.properties`.

Service communication URLs

To ensure the distribution of load across nodes, as well as failover and consistent configuration, `${xxx.services.url}` for each bundle must be configured to point to the load balancer.

URL and IP Address configurations

In the "Clustering Configuration" sections of `deploy.properties`, you will need to configure URLs and IP addresses specific to the nodes.

- Security and primary nodes URLs - used by the replica nodes to synchronize data from the primaries.
- Security replica nodes URLs - used by the primary Security node to push OpenAM configuration changes.
- IP addresses of the Core replica nodes - used by the primary Core node for security reasons, employing an IP address white listing strategy.

Node names

Each bundle node must have a unique name across the installation. The install scripts automatically assign primary nodes the name `primary`. Replica names are specified in two locations:

- In the URL and IP address configuration values of `deploy.properties`. For example, to specify the IP address of the Core replica node named 'replicaNode1' include the property `${replica.replicaNode1}.core.ip.address`.
- On the command line, use the same name for the `-r` option when executing the configure target.

> **Note:** Replica node names may only contain alphanumeric characters (for example, aA-zZ, 0-9).

Pre-installation steps

Before you begin be sure you have executed the following steps.

1. Populated `deploy.properties` with the necessary configurations.

   > **Note:** It is mandatory that the deploy.properties setting `security.valid.goto.resources` is configured. This property must contain at least

one valid URL, which is used for OpenAM redirects. Note that in clustered environment, you can add multiple commas separated valid URLs, eg.

```
http://.another.net://,http://.eo.local://
```

2. Installed Java and Groovy on the target application server.
3. Created the Active Drive and EngageOne databases.
4. Configured the load balancer.

> **Note: Windows Server 2016**
>
> EngageOne Server requires access to Active Drive which is a network shared directory. The default local user accounts do not have access to network resources on Windows Server 2016. It is therefore important to ensure that a access to network resources is available for users running EnageOne services.

Installing the primary nodes

Primary nodes are installed the same way as single nodes. Perform the following:

- Install bundle
- (Optional) validate configuration
- Configure bundle
- Start service

> **Note:** For "Configure bundle" use the node type (`-t`) value `primary` to specify the node as primary.

Execute the following for each bundle.

1. To install the bundle perform the following:

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> install
```

2. (Optional) validate the bundle installation was successful:

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> validate
```

3. To configure the bundle perform the following:

Note: The subsequent "configure" step also performs these validations, and prevents you from proceeding if any configuration properties are not specified correctly.

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> -t primary
configure
```

4. For additional information on starting the service, see the **System Services** appendix.

Installing the replica nodes

Once a primary node of each bundle is running follow the same steps on the additional nodes using the node type (`-t`) of `replica`.

Execute the following steps for each bundle:

1. To install the bundle perform the following command:

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> install
```

2. (Optional) to validate the bundle installation was successful perform the following command:

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> validate
```

3. To configure the bundle perform the following command:

   Note: The subsequent "configure" step also performs these validations and prevents you from proceeding if any configuration properties are not specified correctly.

```
groovy eos.groovy -b <bundle> -p <deploy.properties path> -t replica
-r <replicaNode> configure
```

4. For additional information on starting the service, see the **System Services** appendix.

Notification bundle in a clustered environment

The Notification bundle does not directly support clustering.

There are three options for configuring Notifications in a clustered environment:

1. Install a single instance of the Notification bundle.
   It is recommended that this is installed either on its own server, or on the same server as the primary Core bundle.

   The `notification.services.url` property in `deploy.properties` must point to this server so all message producers (Core and Batch bundles on all nodes) can write to the same queue.

   The advantage of this approach is its simplicity. There is no need to configure ActiveMQ clustering.

   The disadvantage is that there is no resilience in the case of a failure of the server hosting the Notification bundle. If that server is not available, messages will *not* be delivered.

2. Use a standalone, clustered instance of ActiveMQ instead of installing the Notification bundle. This is appropriate for environments with an existing ActiveMQ installation.

   The `notification.services.url` property in `deploy.properties` must point to the externally managed JMS instance.

   The advantage of this approach is that ActiveMQ can be clustered for fail-over and scalability purposes.

   The disadvantage is that some knowledge of ActiveMQ and its cluster configuration is required to set up this environment.

3. Use a standalone, clustered instance of a JMS-compliant message broker other than ActiveMQ, instead of installing the Notification bundle.
   This is appropriate for environments that have an existing message queue infrastructure.

   The `notification.services.url` property in `deploy.properties` must point to the externally managed ActiveMQ instance, and the `notification.connectionFactory.class` property must refer to a custom connection factory class. The class must be packaged into a JAR file, and the JAR file must be installed using the instructions in **Installing and configuring the Notification bundle**.

   The advantage of this approach is that the existing queue infrastructure can be leveraged.

   The disadvantage is that the system relies upon a third party JMS client library that is not tested by Precisely.

8 - Configuration changes

This section describes how to make changes to an existing configuration.

In this section

Reconfiguration steps

The "configure" target of the eos.groovy script may be invoked for the initial install and setup, but also for subsequent configuration changes.

Invoke configure as frequently as necessary to establish your desired configuration. After initial install, applying configuration changes requires individual node downtime. In a clustered environment you may stop the bundle service on each node in isolation, run the configure target, and restart the service to avoid system downtime and user disruption. If the service is running, eos.groovy will detect the running service and exit before applying the configuration.

As a general rule, a configuration change should be applied using eos.groovy to all nodes. The general process is:

1. Update `deploy.properties`.
2. Copy (or make available via a file system share) `deploy.properties` to all nodes.
3. Stop the bundle's service of the target node (see **System Services**).
4. Run the eos.groovy configure target on the node.
5. Start the bundle's service on the node.

Steps 3 through 5 should be repeated for each bundle and node in the system.

Reconfiguration use cases

Some configuration changes are applicable to a subset of the five bundles. This section details common reconfiguration use cases and the specific bundles to which they apply - simplifying the reconfiguration steps. Only the bundles identified need to be reconfigured.

eos database credentials

Many environments require periodic password or credential changes due to security policies. The following properties configure the primary eos database:

- db.host
- db.port
- eo.db.name
- eo.db.username

- eo.db.password

Changes to any of these properties requires a reconfiguration on the following bundles:

- Core
- Composition
- Batch

Flowable database credentials

Similar to the eos database, security policies may dictate periodic password or username changes to the flowable database. The following properties configure this database:

- db.host
- db.port
- flowable.db.name
- flowable.db.username
- flowable.db.password

Changes to any of these properties requires a reconfiguration on the following bundle:

- Core

LDAP credentials

The following system configuration properties are used to access the LDAP server. The DN (Distinguished Name) of an existing directory user must be provided. This user must have permission to execute searches and load user records as all EngageOne directory operations are performed by this user.

- ldap.username
- ldap.password

Changes to any of these properties requires a reconfiguration on the following bundles:

- Security
- Core

System user credentials

The System User is configured by the propertie that follow, these are used to make calls between system components over HTTP (inter-service communications). The user must exist in LDAP and can be considered as a restricted system level user with *system administration* type privileges. This user must belong to the LDAP group specified in the `system.user.group` property:

- system.user.username
- system.user.password

Changes to any of these properties requires a reconfiguration on the following bundles:

- Core
- Composition
- Batch

JVM setting configurations

The Java Virtual Machine (JVM) settings often require tuning as system usage ramps up. Each bundle has its own JVM setting:

- `security.jvm.settings`
- `core.jvm.settings`
- `composition.jvm.settings`
- `conversion.jvm.settings`
- `batch.jvm.settings`
- `notification.jvm.settings`

In this case, only the target bundle must be reconfigured. For example, if you modify `${security.jvm.settings}` execute eos.groovy on the Security bundle servers:

```
groovy eos.groovy -p deploy.properties -t primary -b security configure
```

Asset promotion - Import tab

Asset promotion

Asset promotion allows importing or exporting multiple assets and their configuration, from one EngageOne environment to another, at one time.

Previously, only one asset could be imported or exported at a time.

There are two import types: open import, and promotion bundle.

- Open import – import assets that have been created and published through Designer.
- Promotion bundle - import an asset bundle that was exported through the Asset Promotion Export screen.

  Promotion bundles include assets and asset descriptor files.

The following assets are supported for import:

- Interactive templates
- Non-interactive templates
- Active content

The following roles in EngageOne can use asset promotion:

- Community administrator
- Template manager

The system administrator has access to the Asset Promotion SOAP API.

Configure Import Window

Configuring an Import Window is new functionality introduced in EngageOne version 4.4 Service Pack 3.

This feature gives EngageOne users the ability to run imports at a defined time. For example, imports can be run outside of normal business hours.

- A user can choose to import immediately, or queue the import for the next available time when imports have been defined in the system (the import window has been defined).
- The start time of Import Window and duration is defined in `deploy.properties`, under the key `asset.promotion.import.window.default`.

Rules to create the Import window

```
################################################################################
# Asset Promotion import window (optional)
################################################################################
# Multiple import windows can be configured.
# For example:
#
asset.promotion.import.window.myWindowForWeekDays=MON,TUE,WED,THU,FRI;10:30;180
# asset.promotion.import.window.myWindowForWeekend=SAT,SUN;00:00;600
#
# If there are multiple import windows, they might overlap.
# In this case, the system respects all the import windows.
# Import window functionality works according to the server time where
the Core bundle is installed.
# EngageOne uses the GMT time zone regardless of the server time zone,
the Import window definition should refer to time in GMT.
#  The import window definition consists of three sections separated
with ";"
#
# In the following example:
#  asset.promotion.import.window.default=MON,TUE;14:10;60
# We can distinguish:
#  a) Days section "MON,TUE".
#     Allowed values are: MON,TUE,WED,THU,FRI,SAT,SUN
#  b) Import window start time "14:10".
#     Allowed value range for hours is 00-23, for minutes 00-59.
#  c) Import window length in minutes "60" e.g. 60 is one hour, 1440 is
 24 hours.
#     Allowed value range is 1-1440.
#
# Examples:
#
asset.promotion.import.window.default=MON,TUE,WED,THU,FRI,SAT,SUN;00:00;1440
#  Import window is active all the time.
#  Import window is operational starting at midnight of every weekday,
and lasts for 24 hours
#
#
asset.promotion.import.window.myWindowForWeekDays=MON,TUE,WED,THU,FRI;10:30;180
#  Import window is active on MON,TUE,WED,THU,FRI from 10:30, for 180
minutes
#
#  asset.promotion.import.window.myWindowForWeekend=SAT,SUN;14:40;600
#  Import window is active on SAT and SUN from 14:40, for 600 minutes
#  asset.promotion.import.window.specialWednesday=WED;02:00;30
#  Import window is active on WED from 02:00, for 30 minutes
#
#  asset.promotion.import.window.criticalMondayAndFriday=MON,FRI;03:00;120
#  Import window is active on MON and FRI from 03:00, for 120 minutes
```

```
#
#  Default setting: import window is active all the time
#
asset.promotion.import.window.default=MON,TUE,WED,THU,FRI,SAT,SUN;00:00;1440


asset.promotion.import.window.default=MON,TUE,WED,THU,FRI,SAT,SUN;00:00;1440
```

Asset Promotion clustered configuration

In an environment where the Core bundle is part of a cluster, Asset Promotion is configured by default only on the primary Core node.

- Requests for importing assets are performed one after the other.
- Requests for exporting assets are performed at the same time. Because of this, performance can be improved through load distribution.
  - Assign additional nodes as part of the Asset Promotion cluster.
  - Configure the property `asset.promotion.replica.node.names`, in `deploy.properties`. This property is optional.

Note:

- Nodes that are part of an Asset Promotion cluster should be synchronized using a time synchronization service.
- The system clocks on these nodes must be set to within a second of each other.
- If only the primary Core node is used in Asset Promotion, clock synchronization is not required.

The instructions below describe configuring Asset Promotion on the Core replica nodes.

```
# Asset Promotion cluster configuration
#
# In order for the Core replica node to be part of the Asset Promotion
cluster, set the following property:

asset.promotion.replica.node.names

# Asset Promotion replica node names(optional)
# Define the names of the Core replica nodes to include Core bundles in
 the Asset Promotion service cluster.
# For performance reasons, only two replica names are allowed.
# The primary node is always active.
# The maximum number of Core nodes in an Asset Promotion service cluster
 is 3.
# When executing the bundle configuration script on the replica node,
the value of the '-r' parameter must match the values in these properties.
#
# Note: Nodes that are part of an Asset Promotion cluster should be
synchronized using a time synchronization service.
# The system clocks on these nodes must be set to within a second of
each other.
# If only the primary Core node is used in Asset Promotion, clock
synchronization is not required.
#
```

```
# For example:
# asset.promotion.replica.node.names=replicaNode1
# asset.promotion.replica.node.names=replicaNode1,replicaNode2

asset.promotion.replica.node.names=
```

Import window FAQs

The following are common questions about the Import window:

What is the purpose of the import window?

- Import multiple assets, at one time
- Import large assets (templates, active content)
- Import without impacting system performance
- Import outside of working hours
- Import assets that have been worked on from multiple users
- Import into a production Server (locked down server), during a maintenance window

What is the default Import window definition?

- `asset.promotion.import.window.default=MON,TUE,WED,THU,FRI,SAT,SUN;00:00;1440`

  This Import window is active at midnight every week day, and lasts for 24 hours.

Is there a limit to the number of Import windows defined in `deploy.properties`?

- There is no limit to the number of Import windows defined.

Can import windows overlap?

- Yes. If import windows overlap, the system respects all of them.

  In this case, the system will respect all of the import windows.

  When the import is run, a check is made to verify that the Import window is active. If the window is active the import is processed.

  If any window configuration overlaps, the first Import window is used for the import. Other Import windows are not used.

What should I do to make changes in import window definitions active?

- Restart the Core bundle.

  See **System Services** on page 146 for information on restarting services.

If there are two nodes, one is running with the Import window open all the time.

What is the result if the bundle is configured and run on the second node with the Import window open from 14:00 to 17:00?

- In this case, the Import window will be active from 14:00 to 17:00.
- The Import window definition is global for all nodes in the cluster.

  At bundle startup, all current Import window definition are deleted and new values are read from entries defined in `deploy.properties`.

Clustered environment

How does Import window scheduling work in a clustered environment?

The Import window definition is global for all nodes in a cluster.

At bundle startup, all current import Import window definitions are deleted and new values are read from `deploy.properties`, in `asset.promotion.import.window.default`

Note:

- The Import window *must* be the same on all nodes.
- The same `deploy.properties` file, with an import window definition, must be used on all nodes.

  If a different `deploy.properties` file is used, the *same* settings must be defined for the import window.

Entering invalid values

What happens if wrong values are entered?

For example, if "FOO" is entered as the day, or if the import length is entered as "9999999".

- Running an import bundle configuration or verification an invalid entry will result in an error.

  The error will display with an error description.

  In this example, the error will display the following:

  "Value "FOO" is invalid day definition. Expected value(s) is(are): [MON,TUE,WED,THU,FRI,SAT,SUN]. No space allowed."

  "Value "9999999" is invalid import window length definition. Max value should be 1440 [minutes] which is 24 hours. No space allowed."

  Note: All validation errors must be fixed before a bundle can be installed.

Where will the error message display?

- Errors will display as an output from the console, or in a log file from the Core bundle installation or verification.

  For example:

  [INFO] Validating configuration properties

  [ERROR] One or more properties are invalid. Please fix your entries and re-run the target:

  [ERROR] [asset.promotion.import.window.default = FOO;12:10;9999999] Value "FOO" is invalid day definition. Expected value(s) is(are):[MON,TUE,WED,THU,FRI,SAT,SUN]. No space allowed.

  [ERROR] [asset.promotion.import.window.default = FOO;12:10;9999999] Value "9999999" is invalid import window length definition. Max value should be 1440 [minutes] which is 24 hours. No space allowed.

  [ERROR] Failed to configure 'core' bundle please see log file E:\PB\code\engageone\EngageOneBuild\install\src\main\logs\eos-core-configure-2017-05-22-130048.log

Configuration backups

Before the eos.groovy script applies configuration changes, a backup of the current bundle's configuration is made and placed in `${bundle.install.dir}/conf/backup`. Folders are named with a time stamp format - YYYY-MM-DD-hhmmss.fff. For example, 2020-01-01-235959.59. In addition to the install configuration, the current `deploy.properties` file is placed under `${bundle.install.dir}/conf/backup/deploy.last-configure.properties`. The subsequent invocation of 'configure' will place this file in the last backup.

These backups are created for troubleshooting and may be requested by customer support. Maintain the last backup for rollback purposes. Other backups may be manually purged if desired.

9 - EngageOne Scaling

This section describes the process of scaling the EngageOne installation.

In this section

Scaling the installation

EngageOne Server is scalable horizontally through the addition of nodes and the redistribution of bundles (moving them from one server to another to distribute load). When considering your scaling strategies you should take the following into account:

1. Converting from a single to clustered install requires system downtime. Clustered installs are always recommended for production to provide redundancy and failover.
2. Redistributing Core and Security bundles requires system downtime. However, Composition and Conversion do not. In a scenario where a system containing a combination of bundles has reached its maximum resource consumption, consider moving Composition or Conversion first.
3. Adding additional replica nodes does not require system downtime.

The following sections describe a brief overview of these three use cases.

Migrating a single node install to a clustered install

The following steps outline how to convert a single node install into a clustered install. This use case requires system downtime.

1. Allocate additional servers.
2. Configure your load balancer to point to the new servers.
3. Stop all bundle services
4. Populate the 'Clustered Configuration' section of `deploy.properties`.
5. Re-configure existing bundles as a 'primary' node (repeat for each bundle you are clustering).

```
> groovy eos.groovy -b core -p deploy.properties -t primary configure
```

6. Install and configure new replica nodes on the new servers (repeat for each bundle you are clustering).

```
> groovy eos.groovy -b core -p deploy.properties -t replica -r replica1
 install configure
```

7. Start primary node services.
8. Start replica node services.

Redistributing bundles

This operation assumes you have a clustered environment with multiple bundles installed on the same server. In some scenarios you may find specific bundles are consuming more resources than your hardware allows. Moving them to additional servers is a possible solution.

Core and Security bundles

The Core and Security bundles are both a primary/replica architecture. Moving the primary node to a different server of either of these bundles requires system downtime. In addition, the Core bundle requires the migration of specific template data to the new server. You should avoid moving these bundles if possible. The following steps are outlined for the Core bundle, the steps are generally the same for security.

Attention: Uninstalling the primary Core node will delete the template search index. When moving the primary Core node, you must execute the template target of the migration script on the new node. For more information, see the **Migration Script appendix**.

1. Allocate additional servers
2. Configure your load balancer to point to the new servers.
3. Update the 'Clustered Configuration' section of `deploy.properties`.
4. Stop Core services.
5. Install and configure the new primary (or replica) nodes on the new servers.

```
> groovy eos.groovy -b core -p deploy.properties -t primary install
configure
```

6. Execute the template target of the migration script on the new primary node to re-index your templates (Core only).
7. Start new primary node.
8. (Optional) uninstall old instances.

Composition and Conversion bundles

The Composition and Conversion bundles can be moved without downtime.

1. Allocate additional servers.
2. Configure your load balancer to point to the new servers.
3. Update the 'Clustered Configuration' section of `deploy.properties`.
4. Stop composition services.
5. Install and configure the new primary (or replica) nodes on the new servers.

```
> groovy eos.groovy -b composition -p deploy.properties -t primary
install configure
```

6. Start new nodes.
7. Stop old nodes.
8. (Optional) uninstall old instances.

Adding servers to a clustered install

Adding additional replica nodes to a clustered install is the easiest way to horizontally scale the architecture. It does not require system downtime.

1. Allocate additional servers.
2. Configure your load balancer to point to the new servers.
3. Update the 'Clustered Configuration' section of `deploy.properties`.
4. Install and configure new replica node.

```
> groovy eos.groovy -b core -p deploy.properties -t replica -r
replicaNode2 install configure
```

5. Start a new node.

   Attention: Core bundle limitation

   A new core node will not be able to synchronize data until the primary is re-configured and restarted. Users may experience unexpected behaviors during this period. Anywhere from not finding templates in the Interactive search to 401 errors due to lack of LDAP security data. In the case of Core, after starting the new node, apply the same configuration changes to the primary and restart the service. All other bundles do not require this additional step

10 - Enabling SSO to EngageOne applications

The EngageOne Security bundle can be configured to support single sign-on (SSO) to the EngageOne application.

With SSO enabled, users will not have to re-enter their Windows credentials on the EngageOne login page. Instead, users will be automatically signed in to EngageOne.

In this section

Enable SSO for EngageOne

To enable Windows single sign on to the EngageOne Compose applications:

1. Create a new User account within Active Directory to use as your Kerberos principal.
   a) Check the **This account supports Kerberos AES 256 bit encryption** option on the **Account** tab for the new account.

   See "Create an account in active directory for your Kerberos principal" at https://forgerock.org/2016/08/openam-windows-desktop-sso-deep-dive-part-1/ for detailed instructions.
2. Generate a keytab file for the new account.
   a) From an elevated command prompt, run the following command.

   ```
   ktpass.exe -out openamKerberos.keytab -pass +rndPass -maxPass 256
   -mapuser <accountname> -princ HTTP/<app server host name>@<ACTIVE
   DIRECTORY DOMAIN> -ptype KRB5_NT_PRINCIPAL -kvno 0 -crypto
   AES256-SHA1
   ```

   where:

   - `<accountname>` is the name of the account you created in the previous step
   - `<app server host name>` is the host name of the server running the EngageOne Security bundle (or the host name of the load balancer when clustering Security bundles) in lower case.
   - `<ACTIVE DIRECTORY DOMAIN>` is the Active Directory

   For example:

   ```
   ktpass.exe -out openamKerberos.keytab -pass +rndPass -maxPass 256
   -mapuser openamKerberos -princ HTTP/server1-eoapp.pbeo.net@EO.REMOTE
   -ptype KRB5_NT_PRINCIPAL -kvno 0 -crypto AES256-SHA1
   ```

   b) Copy the resulting keytab file to each of the servers running the EngageOne Security bundle.
3. Configure DNS.

   The user must be able to resolve the EngageOne Security bundle IP address from a DNS forward lookup on the host name. The host name returned from a reverse DNS lookup on the EngageOne Security bundle IP address must match the host name used in the forward lookup.

   See 'Setting up DNS' at: **https://forgerock.org/2016/08/openam-windows-desktop-sso-deep-dive-part-1/** for detailed instructions.
4. Install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files (required to support AES 256-bit cryptography) on the servers running the EngageOne Security bundle.

a) Download JCE from:
http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html
b) Extract the downloaded .zip. Copy `local_policy.jar` and `US_export_policy.jar` to the `lib\security` sub-folder of the Java installation on each server.

5. Re-configure the EngageOne Security bundles to enable SSO:
   a) Edit `deploy.properties`.
   b) Set `security.sso.kerberos.principal.name` to *exactly* (match case) the `-princ` name used in the `ktpass` command in step 2.
   c) Set `security.sso.kerberos.keytab.path` to the location of the keytab file.
   d) Set `security.sso.kerberos.server` to the fully qualified domain name of the Active Directory server.
   e) Configure the Security bundle using the modified `deploy.properties` file.
   For example, run `groovy eos.groovy -b security -p deploy.properties -t single configure`

   For example:

   ```
   security.sso.kerberos.principal.name=HTTP/server1-eoapp.pbeo.net@EO.REMOTE

   security.sso.kerberos.server=domain-controller.pbeo.net

   security.sso.kerberos.keytab.path=c:\\openamKerberos.keytab
   ```

6. Configure browsers used to log on to EngageOne applications to stop prompting for Windows credentials.

   See 'Configuring the web browser' at:
   https://forgerock.org/2016/08/openam-windows-desktop-sso-deep-dive-part-1/ for detailed instructions.

11 - Securing network traffic with TLS

As an enterprise application EngageOne Server leverages several communication paths over the network to transmit information. The following sections describe the methods of security supported by EngageOne Server and the steps to enable them.

In this section

Public / private key archive format

EngageOne Server supports both JKS and PKCS12 Keystore formats. The default is PKCS #12.

PKCS #12 (Public-Key Cryptography Standards 12) is the successor to Microsoft's "PFX" file format and is fast replacing the JKS (Java KeyStore) file format. PKCS #12 files have a .p12 or .pfx filename extension.

A PKCS #12 file is an archive that can store multiple cryptography objects, generally private keys and their X.509 certificates. A PKCS #12 file may also contain any associated chain of trust certificates. Each entry can have a 'friendly' name or 'alias', allowing multiple certificates to be easily recognized and referenced.

PKCS #12 files can be created and manipulated with many tools, including the Java Keytool and OpenSSL pkcs12 command.

> **Note:** For more information on creating the PKCS #12 file, see **Creating PKCS#12 Archives with Java keytool** on page 147 appendix.

Certificate configuration

PKCS support

Ensure you have PKCS12 keystores and truststores for the servers on which EngageOne server componenets are going to be deployed.

Ensure the alias defined for each certificate in the keystore and truststore is unique in the stores, and across the keystore and truststore.

Update `deploy.properties`

- `tls.trust.store.location=<Path to the PKCS Trust store certificate>`
- `tls.trust.store.password=<PKCS Trust store password>`
- `tls.key.store.location=<Path to the PKCS Key store certificate>`
- `tls.key.store.password=<PKCS Key store password>`

JKS support

Ensure you have PKCS12 and JKS keystores and trust stores for the servers on which EOS components are going to be deployed.

Update `deploy.properties`

PKCS entries should exist for Designer to run.

- `tls.trust.store.location=<Path to the PKCS Trust store certificate>`
- `tls.trust.store.password=<PKCS Trust store password>`
- `tls.key.store.location=<Path to the PKCS Key store certificate>`
- `tls.key.store.password=<PKCS Key store password>`

JKS entries co-exist with PKCS entries.

EOCS components: Security, Core, Composition, Conversion, and Notification will use JKS and Designer will continue using PKCS.

- `tls.jks.trust.store.location=<Path to the JKS Trust store certificate>`
- `tls.jks.trust.store.password=<JKS Trust store password>`
- `tls.jks.key.store.location=<Path to the JKS Key store certificate>`
- `tls.jks.key.store.password=<JKS Key store password>`

Converting a JKS Keystore/Trust Store to PKCS

```
keytool -importkeystore -srckeystore <path to jks keystore> -srcstoretype
JKS -destkeystore <path to pkcs keystore> -deststoretype pkcs12
```

Converting a PKCS Keystore/Trust Store to JKS

```
keytool -importkeystore -srckeystore <path to pkcs keystore> -srcstoretype
PKCS12 -destkeystore <path to jks keystore> -deststoretype jks
```

Multiple certificates

If you have more than one certificate (for example, if your servers are in different domains), then you will need to assign an alias (also known as 'friendly name') to each key when creating the archive files. Provide the appropriate alias name for each `${bundle.tls.key.alias}` property in `deploy.properties`. In this way, each bundle can find the correct certificate for the domain of the server where the bundle is installed. If no alias is provided, the first key found in the archive is loaded by the bundle.

> **Note:** If you are using the Design Review and Approval feature you must also include a certificate for the EngageOne Designer server, setting the `designer.tls.key.alias` as appropriate.

Re-configuring certificates

EngageOne Server is set use PKCS12 by default. If EngageOne Server needs to be reconfigured for a different store type, perform the following:

- Update `deploy.properties` to match the new store type.
- Reconfigure all components. For example, Security, Core, Composition, Conversion, Notification.

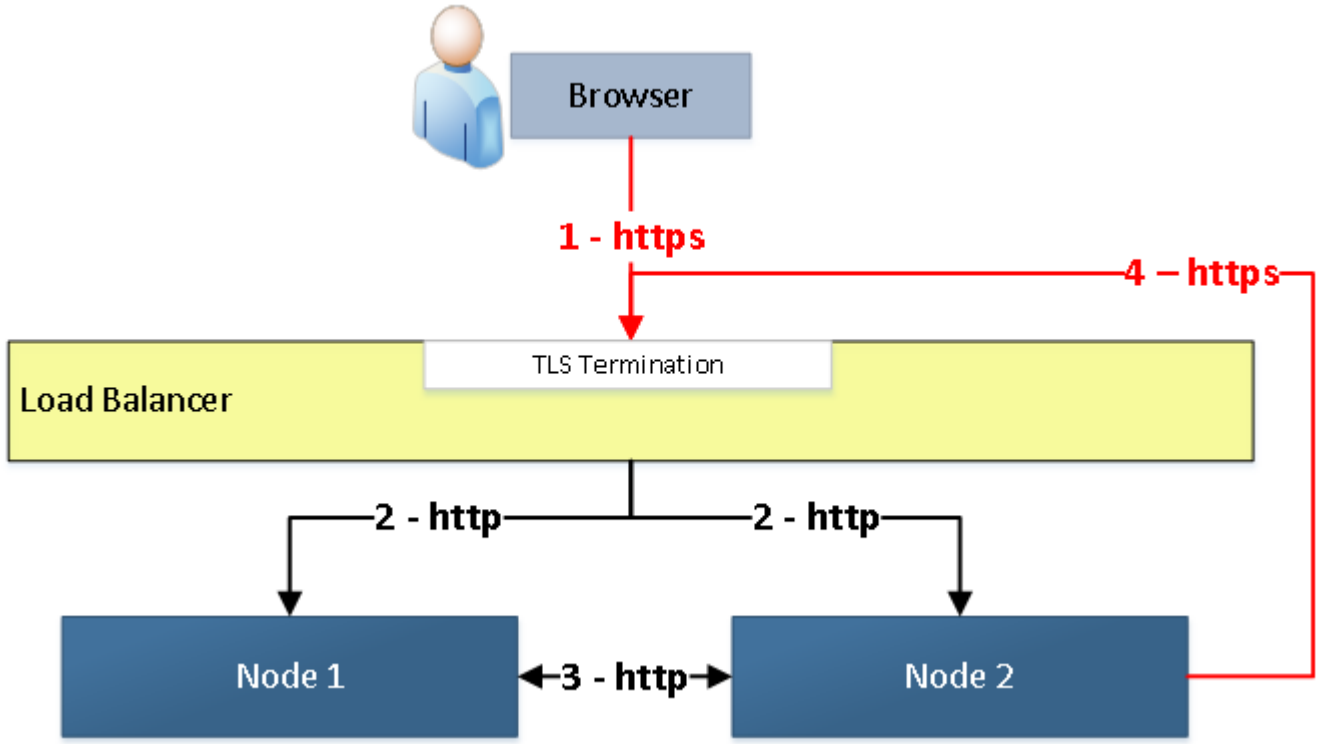
> **Note:** For instructions to reconfigure components see **The eos.groovy script** on page 14.

HTTP protocol

EngageOne Server supports two methods of securing HTTP traffic via TLS and HTTPS. Depending on your security needs you may choose one or the other.

TLS offloading

Below depicts a typical TLS Offloading scenario whereby HTTPS traffic is managed by the load balancer. Here the load balancer communicates with the client using TLS but decrypts the sessions and communicates with the EngageOne Server bundles using HTTP.



TLS offloading configuration - Load Balancer

In addition to the requirements described in **Load Balancer (Clustered Only)** you must also configure:

1. A security certificate with which to perform the TLS encryption and decryption.
2. X-Forwarded-XXX headers to relay the protocol, port and host to the EngageOne Server nodes.

With this configuration EngageOne Server requires identification from the load balancer of the end user's browser connection details. The X-Forwarded-XXX are common headers and are supported by most load balancers.

- `X-Forwarded-Proto` - the protocol (such as HTTPS) with which the load balancer was accessed.
- `X-Forwarded-Port` - the port (such as 80) with which the load balancer was accessed.
- `X-Forwarded-Host` - the host (such as myloadbalancer.com) with which the load balancer was accessed.

These headers are leveraged by EngageOne Server in HTTP redirection cases. For example, when the user is redirected to the login page at session timeout.

TLS offloading configuration - EngageOne Server

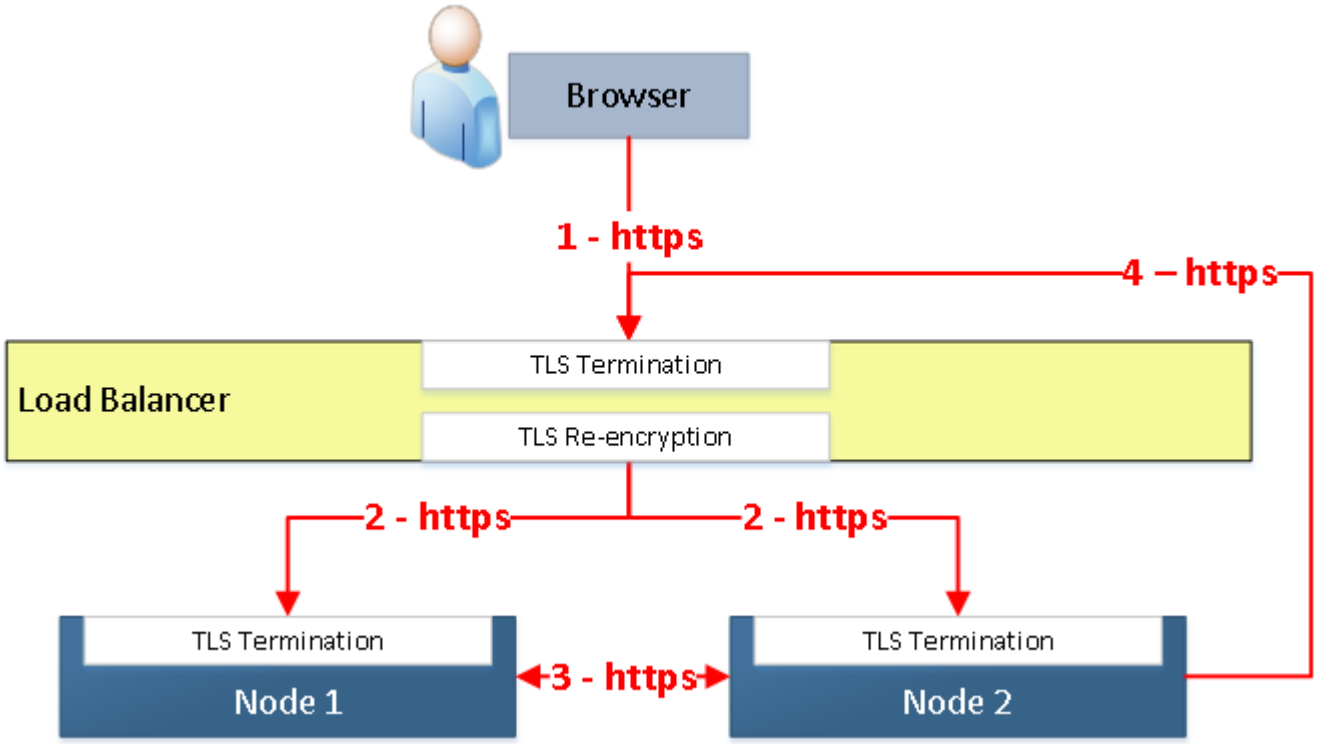
Configuration of the EngageOne Server nodes is minimal:

1. Configure your certificates on each node as described in **Certificate configuration** on page 86.
2. Configure the `${xxx.services.url}` property of each bundle to use HTTPS. For example:
   a) `core.services.url=https://loadbalancer.com:8080`
   b) `conversion.services.url=https://loadbalancer.com:8081`
   c) `security.services.url=https://loadbalancer.com:8082`
   d) `composition.services.url=https://loadbalancer.com:8083`

These URLs are used by EngageOne Server to talk back to itself. These connections should be configured to the load balancer to distribute load and ensure a stable, fault tolerant system.

TLS Re-encryption

Below depicts a typical TLS Re-encryption scenario where HTTP traffic is secure at both the load balancer and application layers. Here the load balancer communicates with the client using TLS (HTTPS), decrypts the sessions so that it can read the payload (cookies and so on), and then re-encrypts the session and communicates with EngageOne Server using TLS (HTTPS).



TLS Re-encryption configuration - Load Balancer

In addition to the requirements described in **Load Balancer (Clustered Only)** you must also configure:

1. A security certificate with which to perform the TLS encryption and decryption.
2. X-Forwarded-XXX headers to relay the protocol, port and host to the EngageOne Server nodes.

> **Note:** See **TLS Offloading Configuration - Load Balancer** for details on the X-Fowarded-XXX headers. This step is optional for re-encryption if the port and host used at the load balancer are the same as the nodes.

TLS Re-encryption configuration - EngageOne Server

To enable encryption at the application level:

1. Configure your certificates on each node as described in **Certificate configuration** on page 86.
2. Configure the `${xxx.services.url}` property of each bundle to use https. For example:
   a) `core.services.url=https://loadbalancer.com:8080`
   b) `conversion.services.url=https://loadbalancer.com:8081`
   c) `security.services.url=https://loadbalancer.com:8082`
   d) `composition.services.url=https://loadbalancer.com:8083`
3. Set the `${xxx.https.enabled}` property of each bundle to `true`. For example:
   a) `core.https.enabled=true`
   b) `conversion.https.enabled=true`
   c) `security.https.enabled=true`
   d) `composition.https.enabled=true`

Ciphers Suite

For https connections the web container uses ciphers suite which is by default configured using ciphers based on recommendations from security bodies (Mozilla, SSL Labs, Oracle) and should be secure enough for most cases and is also compliant with TLSv1, TLSv1.1. TLSv1.2.

Default cipher suite contains the following ciphers:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA

Two optional deploy properties can be used in order to change default ciphers configuration:

- **tls.cipher.suite** - cipher names comma separated list, only the ciphers that are listed and supported by the SSL implementation will be used. The ciphers are specified using the JSSE cipher naming convention.

  Note: that Java does not treat the order in which ciphers are defined as an order of preference unless tls.cipher.suite.useServerCipherSuitesOrder=true is set.

- **tls.cipher.suite.useServerCipherSuitesOrder** - default is false - the first acceptable cipher suite presented by the client will be chosen. Set to true to enforce the tls.cipher.suite cipher order - the list starts with the most preferable ciphers.

You can install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files (required

to support AES 256-bit cryptography) on the servers running the EngageOne bundles.

a) Download JCE from:
http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

Note: Oracle released Java 8 u162. In this version the unlimited policy is enabled by default. You no longer need to install the policy file in the JRE or set the security property

```
crypto.policy
```

JDBC TLS

EngageOne Server supports secured JDBC TLS connections. TLS version depends on database server configuration.

TLS configuration

In deploy.properties configuration file there are two optional properties related to JDBC TLS:

- `jdbc.tls.enabled`
- `jdbc.tls.certificate.subject.dn`

To enable secured JDBC connection set `jdbc.tls.enabled` property to `true`.

Adding certificate to trust store

In most cases database server certificate must be added to EngageOne trust store. Refer to **Certificate configuration** on page 86 for further information.

MS-SQL secured connection can be configured using trustServerCertificate option:

- `trustServerCertificate=true` – connection is established based on database server certificate (less secured option) and you do not need to configure trust store containing server certificate.
- t`rustServerCertificate=false` – connection is established based on certificate stored in EngageOne trust store (more secured option) and you need to add server certificate to trust store. Certificate's subject name or Subject Alternative Name (SAN) must match to server's host name.

Oracle database secured connection always requires to have configured EngageOne trust store with database server certificate.

Certificate subject name

The optional property `jdbc.tls.certificate.subject.dn` allows you to specify server certificate subject DN value and enable additional certificate verification during TLS handshake – more secured option.

If property is set means that:

- MS-SQL connection string contains the `trustServerCertificate=false` option.
- Oracle connection string contains:

```
(SECURITY=(SSL_SERVER_CERT_DN="< jdbc.tls.certificate.subject.dn >"))
```

Used ciphers

EngageOne Server accept any cipher suite configured on database server supported by Java.

SMTP TLS

EngageOne Server supports secured SMTP TLS connections. TLS version depends on SMPT server configuration. STARTTLS and SSL/TLS mode are supported.

TLS configuration

In deploy.properties configuration file there is an optional properties related to SMTP TLS:

- **smtp.security.mode -** allowed values are: `NONE`, `STARTTLS`, `SSLTLS`

Adding certificate to trust store

When `smtp.security.mode` is set to `STARTTLS` or `SSLTLS` then certificate of SMTP server should be added to EO truststore.

Vault TLS

EngageOne Server supports secured connection to Vault over TLS. TLS version depends on Vault server configuration.

TLS configuration

In deploy.properties configuration file there is optional property:

- `vault.tls.enabled`

To enable secured connection to Vault set `vault.tls.enabled` property to `true`.

Trust store / key store

Vault server certificate must be added to EngageOne trust store and key pair to key store. Refer to **Certificate configuration** on page 86 for further information.

Other protocols

In addition to HTTP, EngageOne Server communicates over the following protocols to support various functions:

- LDAP

12 - Configuring external delivery components

In this section

EngageOne Deliver installation

For instructions on installing and configuring EngageOne Digital Delivery, refer to the *EngageOne Digital Delivery Installation Guide*.

Configuration for EngageOne Deliver

In your installation's deploy.properties file locate the `EngageOne Deliver` section found under the `External Delivery System Configuration`. Here you will need to provide the following properties:

- `eodeliver.services.url` - the url to which EngageOne Server can communicate to the EngageOne Deliver services.
- `eodeliver.vendor.identifier` - the name of the vendor (usually 'EngageOne') you have created in EngageOne Digital Delivery.
- `eodeliver.share.dir` - a shared folder between the EngageOne Digital Delivery Server and the EngageOne Server nodes. This folder is established during EngageOne Digital Delivery installation.
- `eo.deliver.admin.url` - the URL of your EngageOne Deliver installation. This `deploy.properties` file setting is located under the `Bundling EngageOne Deliver Administration configuration properties` section. When the URL is provided a button is presented on the EngageOne Administration page allowing you to navigate directly to EngageOne Deliver dashboard.

Stop the Core and Composition bundles and execute the `configure` target on all nodes.

Deliver configuration

Complete the following configuration on the EngageOne Deliver Server. Refer to the *EngageOne Digital Delivery Reference Guide* for full details on how these files are used and configured.

> **Note:** When integrating EO Deliver with EngageOne that is using HTTPS, the following should be added in EO Deliver **deploy.properties** file:

```
eodeliver.jvm.settings=-Xms1024m -Xmx4g -XX:MaxPermSize=512M
eodeliver.https.enabled=true
tls.key.store.location=\\cacerts\\cacerts.pfx
tls.key.store.password=Rkp1WP0eOmh46CO2iJLyW+Lq/nZqpGCdhVkGxkNUu/i8d02tAE1inUg=
tls.key.store.type=PKCS12
```

Configuring the engageone.properties file

Configure the `engageone.properties` file by performing the following steps.

1. Open the EngageOne Deliver `deploy.properties` file with a text editor.
2. Locate the `engageone properties` section.
3. Set the following values:

```
engageone.wsdlURL=https://IP_ADDRESS:PORT/EngageOneWS/RequestStatusUpdateService?wsdl
engageone.domain=EngageOne
engageone.userId=eosuper
engageone.password=qjksLwQ0lm5I2Y0HSCmaRl3tvDemEoyfssoM5OUQxDwmr3iAxsZ8Nik=
```

4. Save, and close the file.

Configuring the servlet.properties file

Configure the `servlet.properties` file by performing the following steps.

1. Go to `<EO Deliver Install directory>\deployments\EODeliver.war\WEB-INF\servlet.properties`
2. Open the `servlet.properties` file with a text editor.
3. Add the EngageOne Server Core bundle's IP addresses to the `outprofile.provider.allowed` property:

   ```
   outprofile.provider.allowed=localhost,127.0.0.1,0:0:0:0:0:0:0:1,${primary.core.ip.address},
   ${replica.replicaNode1.core.ip.address}
   ```

   This property is a security 'whitelisting'. The list of hosts and IP addresses enables those servers to query the outbound profiles. This query is made by the EngageOne Server nodes when you load the EngageOne Admin application's Delivery Channel configuration screen.

4. Save then close the file.

Once you have made these changes to the `engageone properties` (`deploy.properties`) and the `servlet.properties` stop and restart your EngageOne Deliver server.

EngageOne Vault install

For instructions on installing EngageOne Vault, see the *Vault Installation Guide*.

Vault server setup

Before you begin using Vault server you must create or update the `database.ini` and `profiles.ini` files. For more details on this process, refer to the *Vault Installation* or *User's Guide*.

Point EngageOne Server to Vault

Complete the "Vault Configuration (optional)" section, of the "External Delivery System Configuration" section of `deploy.properties`.

Here you will need to provide the following properties:

- `vault.host` - the host with which EngageOne Server should communicate to the Vault render or router (host name or IP).
- `vault.port` - the port with which EngageOne Server should communicate to the Vault render or router (default value: 6003).

Stop the Core and Composition bundles and execute the 'configure' target on all nodes.

Configuring EngageOne Video

This section covers how to configure EngageOne Server to integrate with EngageOne Video.

Point EngageOne Server to EngageOne Video

Here you will need to provide the following properties:

1. Complete the "EngageOne Video Services Configuration (optional)" section of `deploy.properties`.

   You will need to provide the following properties:

   - `video.services.url` - URL to which EngageOne Server can communicate to the EngageOne Video services.
   - `video.services.username` - username used for secure communication with this service.
   - `video.services.password` - encrypted password for the username.
   - `video.services.tenant.id` - ID of the EngageOne Video company to integrate with. Contact the EngageOne Video administrator for this ID.

2. Stop the Core bundle and execute the `configure` target on all nodes.

13 - EngageOne Server uninstall

The following section describes the process of uninstalling EngageOne Server.

In this section

Uninstalling EngageOne Server 4.4

EngageOne Server has a small footprint on any given server or node. Two items are installed on the server:

- Install folder - the install folder of each bundle identified by the `${bundle.install.dir}` property in the configuration file.
- System service - on Windows and Linux a service is installed to manage the start and stop of the software see **System Services** for more detail.

A typical uninstall of the product involves:

1. Shutting down the service of each bundle.
2. Running the `uninstall.groovy` script's `service` target.
3. Running the `uninstall.groovy` script's `bundle` target.

After shutting down the service a typical execution of `uninstall.groovy` looks like the following:

```
> groovy uninstall.groovy -p deploy.properties -b core service
[INFO ] Uninstall 'core' service
[INFO ] Arguments: -b core -p deploy.properties service
[INFO ] Checking eos-core service is not running
[INFO ] Uninstalling eos-core service
[INFO ] Service eos-core has been uninstalled successfully

> groovy uninstall.groovy -p deploy.properties -b core bundle
[INFO ] Uninstall 'core' bundle
[INFO ] Arguments: -b core -p deploy.properties bundle
[INFO ] Validating configuration properties
[INFO ] Checking eos-core service is not running
[INFO ] Status of eos-core service is: uninstalled. Deleting bundle
C:\Program Files\EngageOne\Server\core.
[INFO ] Bundle C:\Program Files\Engageone\Server\core has been deleted
successfully
```

14 - Upgrading EngageOne Server

This chapter provides instructions on upgrading your EngageOne Server environment by the following manual upgrade methods:

- Upgrading from previous 4.4 service packs, refer to **Upgrading from previous 4.4 service packs** on page 106 for detailed information.
- Manually upgrading from previous EngageOne Server versions, refer to **Upgrading from 4.3 to 4.4** on page 112 or, **Upgrading from 3.1.2 to 4.4.x** on page 118, as required for detailed information.

Alternatively, you can use the streamline upgrade method, which allows you to upgrade from any supported version of EngageOne Server. This upgrade method:

- ensures that all upgrade steps including modification of the database structure and active-drive, are performed in the correct order.
- runs the required SQL scripts and groovy migrations automatically.
- notifies you when to manually install the EngageOne Server software.

Note that the streamline upgrade is an alternative upgrade method, you can still use the purely manual upgrade methods outlined in this chapter, as required. Refer to **Streamline upgrade** on page 138 for detailed information.

In this section

Upgrading from previous 4.4 service packs

Introduction

The process of upgrading from a previous 4.4 service pack requires a database schema upgrade and a new software install of the current Service Pack.

System Shutdown and backup

This section describes how to prepare the EngageOne Server for the upgrade procedure.

Stopping network traffic

The EngageOne Server upgrade process requires system downtime and quiet period to execute the purge programs.

Before running any purge process:

- Request all users save their work and log out of EngageOne Interactive.
- Stop all traffic to EngageOne SOAP services.
- Disable network traffic at the load balancer, leaving the system in a running state.

Running purge programs

To ensure a timely upgrade process, execute the EngageOne server purge processes. These applications perform general database and active-drive clean-up. The database schema upgrade may take a considerable runtime if database is not properly purged. Both **purge** and **purge-em** have to be executed, they may take several minutes to run if running them is not already a part of a routine maintenance program.

If there are multiple communities present in an environment about to be upgraded, **purge** has to be executed against each community. No need to do the same with **purge-em**, you only have to execute it once and it should clear Event Monitor for the whole system. For more details on how to run these applications, refer to the EngageOne Administration Guide.

Attention: From EngageOne 4.4.7 onwards, work items are created in their own folder on the active-drive. Prior to this release work items were created in the daily folders on the active-drive. The purge program should be run prior to upgrading to clean up the work items in the daily folders.

There may however be situations where not all work items were in a purgeable state at the time of upgrade; the -workItemInDailyfolders legacy switch is provided to cater for this scenario. It allows the purge program to clean-up any work items remaining in the daily folders.

Shutdown all bundles

After executing the purge programs, shutdown all the EO bundles installed (core, composition, conversion, notification, security). If you are upgrading a clustered environment, stop services for all the nodes.

Backup database

Using the preferred archiving mechanism, backup the EngageOne server database (Microsoft SQL Server or Oracle).

Upgrade preparation

Ensure that you have followed the sections **Extracting the release distribution** on page 5 and **Installing scripting prerequisites** on page 9. The release has to be extracted to a temporary location which will be referred to as `<release-distribution>` throughout this section.

The folder structure of the .zip is as follows:

```
<release-distribution>/
    /bundles
    /docs
    /install
    /samples
    /upgrade
      migrate.groovy
      /active-drive
      /database - see note below.
        /mssql
        /oracle
      /migration - see note below.
        /communities-migrator
        /template-indexer
        /workflow-engine-migrator
        /workflow-migrator
        /workitem-migrator
    /utilities
```

Note the following information regarding the zip file structure shown above:

- `/database` - contains scripts to upgrade SqlServer and Oracle schemas found in the corresponding sub-folders.
- `/migration` - these utilities are required to migrate the existing data structure to a format compatible with the EngageOne Server version being upgraded to.

Database schema upgrade

This section describes how to upgrade your database schema.

Running the upgrade script

There may be one or more upgrade scripts that need to be invoked before the bundles are reinstalled, this will depend on which Service Packs you're performing the upgrade. All the scripts can be found in folder:

```
<release-distribution>/upgrade/database/<mssql or oracle>
```

In this folder there are a series of scripts:

```
<release-distribution>/upgrade/database/<mssql or oracle>/
/mssql
...
/ 06-EngageOne-mssql-4.4.x_to_4.4.3.sql
/ 07-EngageOne-mssql-4.4.3_to_4.4.4.sql
/ 08-EngageOne-mssql-4.4.4_to_4.4.5.sql
etc.
/oracle
...
/ 05-EngageOne-oracle-4.4.x_to_4.4.3.sql
/ 06-EngageOne-oracle-4.4.3_to_4.4.4.sql
/ 07-EngageOne-oracle-4.4.4_to_4.4.5.sql
```

Note: The appropriate scripts must be executed, as detailed below:

- If you are upgrading from EO 4.4.0, EO 4.4.SP1, EO 4.4.SP2, all scripts for either Microsoft SQL Server or Oracle must be executed.
- You must not execute either *06-EngageOne-mssql-4.4.x_to_4.4.3.sql* (for MS SQL Server) or *05-EngageOne-oracle-4.4.x_to_4.4.3.sql* (for Oracle) if you are upgrading from EO 4.4.SP3.
- Only *08-EngageOne-mssql-4.4.4_to_4.4.5.sql* (for MS SQL Server) or *07-EngageOne-oracle-4.4.4_to_4.4.5.sql* (for Oracle) should be executed if you are upgrading from EO 4.4.SP4.

Update active drive

This update is only required when upgrading from EngageOne 4.4 releases prior to the 4.4.Service Pack 8 release.

Changing the Active Drive structure

To change the active-drive structure, run the following:

```
groovy migrate -p deploy.properties active-drive
```

The `migrate.groovy` script is located:

```
<release-distribution>/upgrade
```

This script can be used to migrate or update existing data.

Software installation

When all the previous steps in this section have been completed successfully, you only need re-install each bundle. It is important to reuse the existing active-drive rather than creating a new one.

When the software install and configuration is complete, start the appropriate services; you must then validate installation of each bundle as described in sections **EngageOne standalone install** on page 51 or **EngageOne clustered install** on page 61 (depending on your environment).

Update user rights and system administrators

This update is **only** required when upgrading from EngageOne 4.4 releases prior to the 4.4.Service Pack 8 release.

To update user rights and system administrators, run the following:

```
groovy migrate -p deploy.properties distinguishedName
```

The migrate.groovy script is located:

```
<release-distribution>/upgrade
```

Note that this script can be used to migrate or update existing data.

Template migration

This migration is recommended when upgrading from any other EngageOne 4.4 releases. Template migration will re-index all existing imported templates and can fix certain problems related to marking invalid templates in the Admin module.

The migrate.groovy script is located under <release distribution>/upgrade. Automation of data migration steps can be performed as follows:

```
> groovy migrate.groovy -p deploy.properties template
```

Migrate workflow definitions and tasks

It is important to note that where existing workflows *steps* have been approved either from Review and Approval or Projects, these may need re-approval after workflow migration in the following scenarios:

- The workflow consists of several review steps in which one or more have been approved.
- In a one-step workflow where there are multiple reviewers assigned to the step and all reviewers have issued approval (**Everyone reviews**) Where one or more reviewers have issued approval; these require re-approval.
- In a one-step workflow where all group member should approve (**Everyone in the group reviews**) . Where one or more reviewers have issued approval; these require re-approval.

You need to run the workflow migration detailed in this section when

- Upgrading from EngageOne 4.x to 4.4.10 and above and,
- only if workflows have been defined for Projects or Review and Approval.

You do not need to run the migration task if you have workflows set up in pre 4.4.10 versions but do not wish to carry them forward,

To migrate workflow definitions and tasks run:

```
groovy migrate -p deploy.properties workflow-engine
```

The migrate.groovy script is located:

```
<release-distribution>/upgrade
```

Note: From EngageOne 4.4 ServicePack 9 to EngageOne 4.4 ServicePack 10 the following properties in deploy.properties files were changed from:

```
activiti.db.name=
activiti.db.username=
activiti.db.password=
```

to:

```
flowable.db.name=
<when migrating value should be the same as it was for activiti.db.name>
flowable.db.username=
<when migrating value should be the same as it was for
activiti.db.username>
flowable.db.password=
<when migrating value should be the same as it was for
activiti.db.password>
```

Note: Before running migrate script make sure that security bundle and core bundle are up and running.

Migrate custom code pages

This migration is only required when upgrading from EngageOne 4.4 releases prior to the 4.4 ServicePack 10-P1 release.

To migrate custom code pages, run the following:

```
groovy migrate -p deploy.properties customCodePage
```

The migrate.groovy script is located:

```
<release-distribution>/upgrade
```

Note: This script can be used to migrate or update existing data.

Upgrading from previous EngageOne versions

Upgrading from 4.3 to 4.4

Introduction

This guide describes the process to upgrade EngageOne Server from 4.3 to 4.4. This process requires a new software install of the 4.4 system, followed by a data migration against the running system.

Software and hardware requirements

Please note that supported Java version has changed between EngageOne 4.3.X and EngageOne 4.4.X.

Refer to the *EngageOne 4.4 Software and Hardware Requirements* to see the full list of requirements before performing the upgrade.

Upgrade preparation

On the server which you intend to execute the migration utilities (primary Core bundle is recommended), ensure that you have followed the sections **Extracting the release distribution** on page 5 and **Installing scripting prerequisites** on page 9.

You should now have the release extracted to a temporary location which will be referred to as `<release-distribution>` throughout this document.

The contents of the .zip that pertain to the upgrade are as follows:

```
<release-distribution>/
    /bundles
    /docs
    /install
    /samples
    /upgrade
      migrate.groovy
      /active-drive
      /database - see note below.
        /mssql
        /oracle
      /migration - see note below.
        /communities-migrator
        /template-indexer
```

```
        /workflow-engine-migrator
        /workflow-migrator
      /workitem-migrator
    /utilities
```

Note the following information regarding the zip file structure shown above:

- `/database` - contains scripts to upgrade SqlServer and Oracle schemas found in the corresponding sub-folders.
- `/migration` - these utilities are required to migrate the existing data structure to a format compatible with the EngageOne Server version being upgraded to.

System Shutdown and backup

This section describes how to prepare the EngageOne Server for the upgrade procedure.

Stopping network traffic

- Request all users save their work and log out of EngageOne Interactive.
- Stop all traffic to EngageOne SOAP OnDemand SOAP services.
- Ideally, disable all network traffic at the load balancer leaving the system in a running state.

Shutdown application bundles

Stop all EngageOne services.

Backup database and installation

Using your preferred archiving mechanism, backup the EngageOne Server database (SQLServer or Oracle), and the EngageOne Server installation folder.

Backup the installation folder in a .zip, or .tar archive as it may be necessary to reference it while migrating common custom configurations.

Deployment configuration upgrade

All information required to deploy EngageOne is held in your existing `deploy.properties` file.

It is recommended that you copy your existing `deploy.properties` file to create the configuration for the deployment of the new version of EngageOne.

In your new `deploy.properties`:

1. Modify existing settings to reflect any changes you wish to make in your deployment of the new version of EngageOne.

   For example, server name changes, password changes, etc.

2. Add the properties required for your new version of EngageOne Server, refer to the EngageOne_4.4_Configuration_Checklist document for detailed information.

Database schema upgrade

Update the database schema using the sequence of .sql upgrade scripts found in `<release-distribution>/upgrade/database/<mssql or oracle>`.

In this folder you will find a series of scripts:

```
<release-distribution>/upgrade/database/<mssql or oracle>/
   /mssql
      ...
      /05-EngageOne-mssql-4.3.0_to_4.4.0.sql

      etc.

   /oracle
      ...
      /04-EngageOne-oracle-4.3.0_to_4.4.0.sql
         and so on.
```

Copy these scripts to a temporary location on the target database server. Execute them in the sequence outlined in the file names, for 4.3_to_4.4.

Software install

Keep the following in mind while performing the install:

- Make sure to reuse the existing active drive and do not create a new one.

  Subsequent steps in this guide will make any necessary adjustments to the active drive.

Once the software install and configuration is complete, start the services.

Though the system may appear to be functioning at this point, be sure to complete the Data Migration steps below to ensure the integrity of your data.

Data migration

The following steps migrate various components in the 4.3 database to a 4.4 format.

Perform these steps from the server on which the Core bundle primary (or single) node is installed.

Before you begin:

- Verify the Core bundle is running by visiting the 'status' application - `http://<server>:<port>/status`
- Verify the server has the correct version of java available in the shell PATH.

  For more information, see the *EngageOne 4.4 Software and Hardware Requirements Guide*.

Template migration

The `migrate.groovy` script is located under `<release distribution>/upgrade`. This can be used to automate required data migration steps:

```
> groovy migrate.groovy -p deploy.properties template
```

For more details how to use `migrate.groovy` script please see the "Migration script" section below, or run the `groovy migrate.groovy -h` command to view help information.

Migrate workflow definitions and tasks

It is important to note that where existing workflows*steps* have been approved either from Review and Approval or Projects, these may need re-approval after workflow migration in the following scenarios:

- The workflow consists of several review steps in which one or more have been approved.
- In a one-step workflow where there are multiple reviewers assigned to the step and all reviewers have issued approval (**Everyone reviews**) Where one or more reviewers have issued approval; these require re-approval.
- In a one-step workflow where all group member should approve (**Everyone in the group reviews**) . Where one or more reviewers have issued approval; these require re-approval.

You need to run the workflow migration detailed in this section when

- Upgrading from EngageOne 4.x to 4.4.10 and above and,
- only if workflows have been defined for Projects or Review and Approval.

You do not need to run the migration task if you have workflows set up in pre 4.4.10 versions but do not wish to carry them forward,

To migrate workflow definitions and tasks run:

```
groovy migrate -p deploy.properties workflow-engine
```

The migrate.groovy script is located:

`<release-distribution>/upgrade`

> **Note:** From EngageOne 4.4 ServicePack 9 to EngageOne 4.4 ServicePack 10 the following properties in deploy.properties files were changed from:

```
activiti.db.name=
activiti.db.username=
activiti.db.password=
```

to:

```
flowable.db.name=
<when migrating value should point
```

```
the same as it was for activiti.db.name>
flowable.db.username=
<when migrating value should point
the same as it was for activiti.db.username>
flowable.db.password=
<when migrating value should point
the same as it was for activiti.db.password>
```

Note: Before running migrate script make sure that security bundle and core bundle are up and running.

Migration notes

- After the upgrade, you must change the name of any communities that contain the following characters: . & < > ? / \

  These characters are not supported for community names in EngageOne 4.4.

- Template migration is the only target needed for the 4.3 - 4.4 upgrade.

  To confirm template data are migrated, you can perform a template search in EngageOne Interactive.

Migration script

An additional script, `migrate.groovy` is found in the `<release-distribution>/upgrade` folder.

This may be used to migrate existing data. It is a target-based interface allowing you to perform different operations with different invocations.

Below is the usage statement of this script:

```
usage: groovy migrate.groovy [options] [target [target2 [target3]...]]
 -h,--help                Usage information.
 -l,--logback <arg>       Specify the location of the logback
                          configuration file.
 -p,--properties <arg>    Specify the location of the properties file
                          containing configuration information.
targets:
 active-drive    -> Update active drive structure.
 community       -> Migrate the viewpoint communities.
 template        -> Migrate the EngageOne templates.
 workflow        -> Migrate workflow definitions and instances from
                    EO 3.2.1 to 4.4.10+
 workflow-engine -> Migrate workflow definitions and instances from
                    EO 4.x to 4.4.10+
 workitem        -> Migrate work items.
```

The properties file shares a subset of the configuration data as documented for `deploy.properties`. This file must adhere to all requirements for the new software installation.

By default, migration tools run with logging level set to WARN.

To change this, specify the location of a logback XML configuration file, and increase the console logging level.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <shutdownHook class="ch.qos.logback.core.hook.DelayingShutdownHook"/>
 <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
   <level>INFO</level>
  </filter>
  <encoder>
   <pattern>[%date | %-5level | %1logger] %msg%n</pattern>
  </encoder>
 </appender>
 <root level="INFO">
  <appender-ref ref="console"/>
 </root>
</configuration>
```

Troubleshooting

If a template search does not work after an upgrade is performed (including performing template migration):

Ensure active-drive, and `<Eos Installed dir>/ bundles\core\conf\viewpoint-indexing` have full access.

1. Go to **EO Admin > Templates**
2. Select the template folder where search does not work:
   a. Right-click and select **OK**.

Upgrading from 3.1.2 to 4.4.x

Introduction

This guide describes the process to upgrade EngageOne Server 3.1.2 to 4.4.x.

This process requires a database schema upgrade and a new software install of the 4.4.x system, followed by a data migration against the running system.

Software and Hardware requirements

Please refer to the *EngageOne 4.4 Software and Hardware Requirements* before performing the upgrade.

Upgrade preparation

On the migration server (primary Core bundle is recommended), ensure that you have followed the sections **Extracting the release distribution** on page 5 and **Installing scripting prerequisites** on page 9.

You should have the release extracted to a temporary location, referred to as `<release-distribution>`.

The contents of the .zip that pertain to the upgrade are as follows:

```
<release-distribution>/
    /bundles
    /docs
    /install
    /samples
    /upgrade
      migrate.groovy
      /database - see note below.
        /mssql
        /oracle
      /migration - see note below.
        /communities-migrator
        /template-indexer
        /workflow-engine-migrator
        /workflow-migrator
        /workitem-migrator
    /utilities
```

Note the following information regarding the zip file structure shown above:

- `/database` - contains scripts to upgrade SqlServer and Oracle schemas found in the corresponding sub-folders.

- `/migration` - these utilities are required to migrate the existing data structure to a format compatible with the EngageOne Server version being upgraded to.

Configuring EngageOne active drive

If you have an existing installation configured using mapped network drives, before upgrading to EngageOne 4.4.x you must re-configure EngageOne to use the UNC share path instead of the drive mapping.

To do this perform the following steps:

1. Log in to EngageOne Administration.
2. Navigate to **System Administration** > **Configuration Management** > **Disk Configuration**
3. Change **Shared Path** to the equivalent UNC path.

At this point EngageOne will update the relevant data to replace the drive mapping references with the UNC path references.

System Shutdown and backup

This section describes how to prepare the EngageOne server for upgrade.

Stopping network traffic

The EngageOne Server upgrade process requires system downtime and quiet period to execute the database purge programs.

Before running any purge process:

- Request all users save their work and log out of EngageOne Interactive.
- Stop all traffic to EngageOne SOAP OnDemand SOAP services.
- Disable network traffic at the load balancer, leaving the system in a running state.

Executing purge programs

To ensure a timely database upgrade and migration process, execute the EngageOne server purge programs.

These applications perform general database and active-drive cleanup. These applications may take several minutes to run if this is not already part of a routine maintenance program.

For details on running these applications, refer to the *EngageOne Server 3.1.2 Administration Guide*.

Execute purge

The purge application for each community (previously 'domain') is used to clean up tasks (previously 'work items') with a maximum numDaysRetention of 7 (one week).

By default, purge will delete completed and failed tasks.

To modify this behavior please see the *EngageOne Server 3.1.2 Administration Guide*.

```
purge.bat -domain community1
purge.bat -domain community2
and so on.
```

Execute purge-em

The purge-em application deletes records in the EventMonitor database.

It is critical to remove all EventMonitor records that are older than 7 days (one week) from the database.

The database schema upgrade performs a data transformation on these tables which will require a considerable runtime if this data is not properly purged.

Set the onOrBefore option to no sooner than one week before "today".

```
purge-em.bat -onOrBefore <yyyy-MM-dd>
```

Shutdown application server

After executing the purge programs, shutdown the application server or servers.

Backup database and installation

Using the preferred archiving mechanism, backup the EngageOne server database (SQLServer or Oracle), and the EngageOne server installation folder.

Backup the installation folder in a .zip or .tar archive as it may be necessary to reference it while migrating common custom configurations.

Upgrading the database schema

Update the database schema using the sequence of .sql upgrade scripts found in `<release-distribution>/upgrade/database/<mssql or oracle>`.

In this folder you will find a series of scripts:

```
<release-distribution>/upgrade/database/<mssql or oracle>/
   /mssql
      /3.1.2_to_4.0
         00-EngageOne-mssql-3.1.2_to_4.0.4.sql
         01-EngageOne-mssql-3.1.2_to_4.0.4.sql
         etc.
      /4.0_to_4.1
         00-EngageOne-mssql-4.0_to_4.1.0
         etc.
      /4.1_to_4.2
         etc.
      /4.2_to_4.3
```

```
        etc.
   /oracle
     etc.
```

Copy these scripts to a temporary location on the target database server. Execute them in the sequence outlined in the file names.

Software install

Install the EngageOne server software on your existing hardware or new hardware.

Keep the following in mind while performing the install:

- Make sure to reuse the existing active-drive and do not create a new one.

  Subsequent steps in this guide will make necessary adjustments to the active-drive.

- The existing database from 3.1.2 is equivalent to the 'eos' database.
- A new database 'flowable' is required in 4.4.x that was not present in 3.1.2.

Once the software install and configuration is complete, start the servers.

The system may appear to be functioning at this point, however be sure to complete the data migration steps to ensure the integrity of your data.

Data migration

The following steps migrate various components in the 3.1.2 database to a 4.4.x format.

Migration script

An additional script, `migrate.groovy` is found in the `<release-distribution>/upgrade` folder and may be used to migrate existing data.

It is a target-based interface allowing you to perform different operations with different invocations.

> **Note:** If communities will be migrated through the `migrate.groovy` script by executing `migrate.groovy -p deploy.properties community`, the following must exist:
>
> - The user specified in `deploy.properties` **must** have system administrator privileges.
>
> The *same* user name must be entered in `initial.sysadmin.username,` and `system.user.username`
>
> Instructions for configuring system users and system administrators are detailed in the "Administration guide", and in `deploy.properties`.

Below is the usage statement of this script:

```
usage: groovy migrate.groovy [options] [target [target2 [target3]...]]
 -h,--help               Usage information.
```

```
 -l,--logback <arg>      Specify the location of the logback
                         configuration file.
 -p,--properties <arg>   Specify the location of the properties file
                         containing configuration information.
targets:
 active-drive    -> Update active drive structure.
 community       -> Migrate the viewpoint communities.
 template        -> Migrate the EngageOne templates.
 workflow        -> Migrate workflow definitions and instances from
                    EO 3.2.1 to 4.4.10+
 workflow-engine -> Migrate workflow definitions and instances from
                    EO 4.x to 4.4.10+
 workitem        -> Migrate work items.
```

The properties file shares a subset of the configuration data as `deploy.properties` . These properties must adhere to the requirements for the new software installation.

By default, migration tools run with logging level set to WARN. To change this, specify the location of a logback xml configuration file and increase the console logging level.

For example:

```
<?xml version="1.0" encoding="UTF-8"?><configuration>
 <shutdownHook class="ch.qos.logback.core.hook.DelayingShutdownHook"/>
 <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
   <level>INFO</level>
  </filter>
  <encoder>
   <pattern>[%date | %-5level | %1logger] %msg%n</pattern>
  </encoder>
 </appender>
 <root level="INFO">
  <appender-ref ref="console"/>
 </root>
</configuration>
```

Script targets

The targets should be run after the database schema has been upgraded, the EngageOne Server 4.4.x Core bundle installed and services started.

These should be run in the order:

1. active-drive
2. community
3. workflow
4. workitem
5. template

These targets depend on the following properties configured in `deploy.properties`:

```
# Database server type.
db.type=


# Name of the SQL or Oracle database server.
db.host=


# Port on which to connect to the database server.
db.port=


# EngageOne database
eo.db.name=


# EngageOne database user
eo.db.username=


# EngageOne database user password.
eo.db.password=


# Installation directory
core.install.dir=
```

The following steps migrate various components in the 3.1.2 database to a 4.4.x format.

Perform these steps from the server where the primary Core bundle (or single) node is installed.

Before you begin:

- Verify the Core bundle is running by visiting the 'status' application - `http://<server>:<port>/health-check-web`
- Verify the server has the correct version of Java available in the shell PATH.

  See Software Requirements in the *EngageOne 4.4 Hardware and Software Requirements Guide*.

Run migration

The `migrate.groovy` script is located under `<release distribution>/upgrade` and can be used to automate the data migration steps:

```
> groovy migrate.groovy -p deploy.properties community
> groovy migrate.groovy -p deploy.properties template
> groovy migrate.groovy -p deploy.properties workflow
> groovy migrate.groovy -p deploy.properties workitem
> groovy migrate.groovy -p deploy.properties active-drive
```

If you are not using the 3.1.2 workflow engine, you may skip running the migration script against 'workflow' target. All target data migration steps may be executed in a single invocation:

```
> groovy migrate.groovy -p deploy.properties community template workflow
 workitem active-drive
```

Each script target automates execution of the appropriate migrator application and records any output to a separate log file.

These log files are currently located under `<release-distribution>/install/logs`.

> **Note:** Workflow migration assumes that all active workflow definitions and workflow task instances are to be migrated.
>
> The Workflow Migration Application supports a number of alternative modes of operation, and there may be a desire to run this application separately.

Active-drive migration

Active-drive structure contains `<active.drive.dir>/config/config-settings.xml` configuration file which is no longer used.

In version 4.4.x, each bundle has its own local `config-settings.xml` in the `<bundle.install.dir>/conf/config-settings/` folder.

Verify migration

Use the following sections to verify your migration once complete.

Verify community migration

After running the `groovy.migrate` command for the community component, you should see the following message:

```
community component has been successfully migrated
```

Once the migration is complete the communities and roles as configured in the 3.1.2 install are available in the new EngageOne Administration interface.

1. In a browser, navigate to the Core bundle - `http://<server>:<port>`
2. Login as the initial system administrator as configured by `${initial.sysadmin.username}` at install time.
3. Click the **Admin** tab.
4. Click **Communities** and review the migrated communities.
5. For each community, click **Role Mapping** and review the migrated permissions. To switch communities use the drop-down in the upper right corner.

Verify template migration

After running the `groovy.migrate` command for the templates you should see the following message:

```
templates have been successfully migrated
```

Once migration is complete the templates will be fully indexed.

Verify execution through the EngageOne Interactive application:

1. In a browser, navigate to the Core bundle - `http://<server>:<port>`
2. Login as the initial system administrator as configured by `${initial.sysadmin.username}` at install time.
3. Select the community in the upper right of the window.
4. Click the **Templates** tab.
5. Click the search icon (magnifying glass).
6. Enter a search of a known existing template
7. Repeat for each community.

Verify work items migration

After running the `groovy.migrate` command for the work items you should see the following message:

```
Work items have been successfully migrated
```

Verifying workflow migration

After running the `groovy.migrate` command for the workflow you should see the following message:

```
Workflow has been successfully migrated
```

Once migration is complete, the compatible workflow definitions and associations plus any active workflow task instances will have been migrated from EDMS to flowable.

Verify execution through the EngageOne Interactive application:

1. In a browser, navigate to the Core bundle - `http://<server>:<port>`
2. Login as the initial system administrator as configured by `${initial.sysadmin.username}` at install time.
3. Select the community in the upper right.
4. Click the **Admin** tab.
5. Click **Review and Approval Workflows**.
6. Verify known existing workflow definitions are present.
7. Click **View workflow associations**.
8. Verify the anticipated associations are present.

9. Click the **Task** tab.
10. Verify existing active tasks are present.
11. Repeat for each community.

Appendix

Workflow migration application

In EngageOne version 4.4.x, the previous workflow engine (“OpenEDMS”) was replaced with a new, standards-based engine (“flowable”).

The workflow design environment, storage location and execution engine have changed. The workflow migrator application tool has been created to enable a one-time migration of OpenEDMS workflows to flowable workflows.

The workflow migrator application retrieves Template metadata from the EngageOne server. The workflow migrator application is located under `<release distribution>/upgrade/utilities/workflow-migrator`.

Execute the following steps directly from this folder.

Workflow migration application overview

The purpose of the migration tool is:

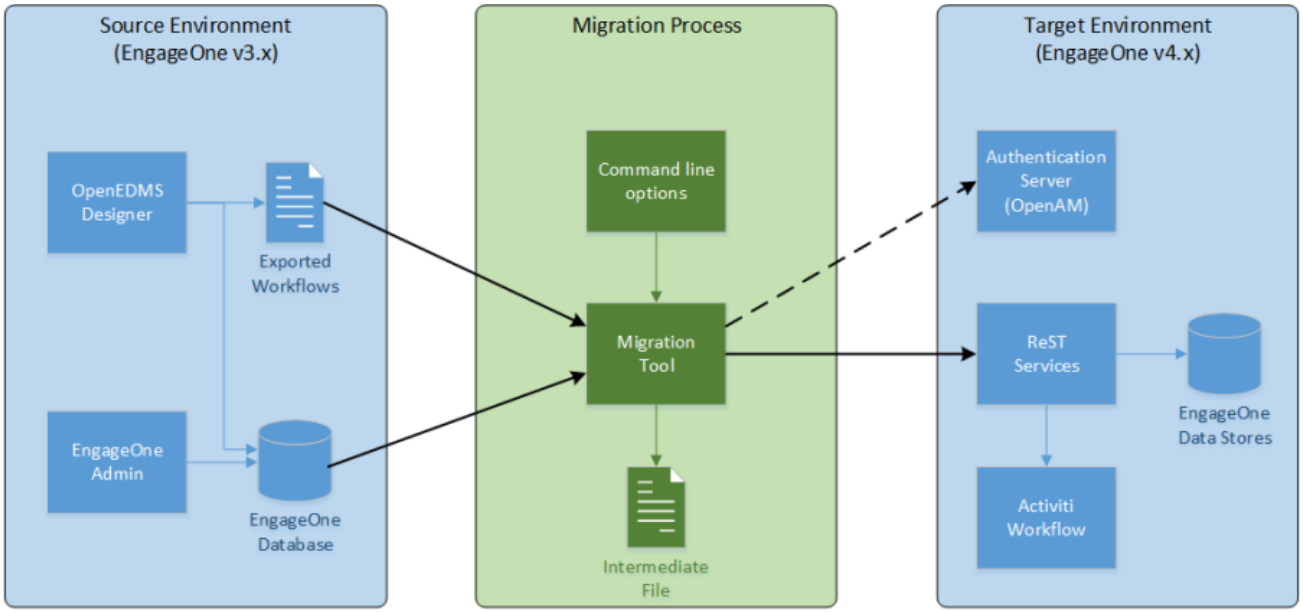
- Read OpenEDMS workflow definitions and in-flight instances from an EngageOne server 3.x database
- Create functionally equivalent definitions for the 4.4.x workflow system
- Save these definitions in the target environment
- Start instances of the new definitions for the in-flight work items

The tool offers different options to provide flexibility in its use.

The main users of the tool will be Professional Services personnel, and customer IT departments involved in upgrading an existing installation of EngageOne server to the latest version.

The tool is used as part of the upgrade process.

The following diagram shows how the migration tool fits into the migration picture:



Source environment:

- Workflow administrators create workflow definitions through the OpenEDMS designer.

  These definitions are saved to the EngageOne database where they are used by the runtime system.

  Optionally, the administrator can export workflow definitions to files.

- EngageOne administrators associate workflow definitions with document classes.

  Document templates can be assigned to a document class, resulting in the workflow being executed when a document is created from that template.

Migration process:

- A user invokes the migration tool through a command prompt. and specifies a set of options through command-line flags.
- The migration tool reads OpenEDMS workflow definitions from either a file (or directory of files), or from the EngageOne database.
- Optionally, the workflow definitions can be written to an intermediate file so that the import into the target environment can be carried out at a different time or place than the export from the source environment.
- The migration tool authenticates the credentials provided as command-line arguments by calling the authentication server.
- Once authenticated, the migration tool invokes a set of web services in the target environment to create the workflows and associate them with users, templates and folders.
- Once workflow definitions have been created in the target environment, in-flight work item records are read from the EngageOne database and instances of the new definitions are started.

Target environment:

- The authentication server is a standard feature of the EngageOne server 4.4.x environment. It accesses user information stored in an LDAP repository.
- REST services provide an internal (unpublished) interface in the 4.4.x environment.

  The migration tool accesses the admin-services, workflow-services and EngageOne REST service.

  The tool assumes these services are located on the same server.

- The flowable server is a standard feature of the 4.4.x environment. It is not directly accessed by the migration tool, but it needs to be available as workflow-services need to access it.

  Note: The tool has a "validate-only" mode.

  It is strongly recommended that this mode is used to identify potential issues that would prevent a successful migration.

  Once any issues have been identified, and potentially resolved, the tool can be re-run to perform the migration process.

Using the workflow migration tool

There are three basic modes of operation for the migration tool:

1. **Export only**

   The tool reads workflow definitions from either a database or files and writes all of the required information to an intermediate file.

   Typically, this mode would be used when concurrent access to the 3.x EngageOne database and the 4.4.x web services is not available.

   After exporting the data to the intermediate file, that file can be copied to another environment where access to the web services is available.

2. **Import only**

   The tool reads the definitions from an intermediate file and calls the 4.4.x web services to create the new definitions.

   This mode is used after the "export" mode has previously been used to generate the intermediate file.

3. **Both (export followed by import)**

   The exports and imports are combined into a single operation.

   Optionally, an intermediate file can be generated, but this will have little benefit.

   "Both" mode has to be used when migrating in-flight instances.

Domains and communities

The concept of domain in EngageOne server 3.x has been replaced by community in 4.4.x.

By default the migration tool processes workflows in all domains and re-creates them in the corresponding community of the target system.

It is possible to influence this behaviour via the "community" flag, as follows:

- When exporting definitions from a database, a single domain name can be specified (such as `-c MyDomain` or `--community MyDomain`).

  The migration will process workflow definitions and associations within that domain.

- When processing definitions from OpenEDMS export files, a domain name can be specified as above, but has a slightly different meaning in this context.

  The OpenEDMS export file contains a domain ID for the workflow but no domain name.

  When a "community" flag is specified in the command line, the migration tool will attempt to re-create workflows in the community, in the target environment.

  If the domain ID and the community name match a community in the target system, the process is proceeding well.

  If the community name exists but does not have the same ID as the original domain, the import will proceed but a warning will be issued.

  If no community is specified in the command line, the migration tool will attempt to look up a community using the domain ID.

  If a matching community is found, workflows will be imported to that community.

  If no matching community is found, workflows will not be imported.

  > **Note:** When an EngageOne system is upgraded from 3.x to 4.4.x, each domain is migrated to a community with the same name and ID as the original domain.

Associations

In the EngageOne server 3.x system, a workflow process can be associated with a template, or a folder of templates.

In either case, the association is managed via a document class.

When the migration tool reads the workflow definitions from an EngageOne database it can (optionally) read these associations and re-create corresponding associations in the target environment. If the association information is not available when OpenEDMS definitions are read from files, the tool is unable to recreate associations in the target environment.

By default, the migration tool does not create associations in the target system. Associations are created if the "associate-template" or "associate-folder" flags are used.

Validation

If the migration tool has access to the target environment. it will attempt to validate data when processing workflow definitions and associated entities.

In "import" or "both" mode, the tool must have access to the target system, to process validation.

In "export" mode, the tool optionally has access to the target system, to process validation if the target server information is provided on the command line.

Note: If "export" and "import" are used separately, validation can be processed twice.

The following items are validated against the target environment:

- **Domain / Community**

  These should be matched on both ID and name, but a partial match on name is acceptable.

- **Users and Groups**

  Users and groups specified in the OpenEDMS workflow definitions must be available in the target environment.

  These are validated on a combination of name and type (where type is user or group).

- **Folders and templates**

  If "associate-folder" or "associate-template" flags are specified, each of the folders or templates associated with the OpenEDMS workflow (via a document class) are validated in the target environment.

  Folders are validated by path.

  Templates are validated by ID (which refers to a version of the template).

Command line options

The following table lists the available options that can be specified on the command line.

All options can be specified using either a short flag (one letter preceded by a dash, for example "-m"), or a longer flag (one or more words preceded by two dashes, for example "--mode").

Some options require additional information after the flag. If the additional information contains a space character, the string should be enclosed in quotes, (for example: `--intermediate-file "c:\my files\intermediate.txt"`).

Flag	Long form	Description
-m	`--mode`	Mode of operation for the migration tool (no default value). • export Workflow definitions are exported from a database or XML files, and stored in an intermediate to be later imported into a 4.4.x system. • import Workflow definitions are read from an intermediate file and imported to a 4.4.x system.

Flag	Long form	Description
		• both Workflow definitions are exported from a database or XML files, and imported into a 4.4.x system. Example: `-m both`
-f	`--intermediate-file`	Name (and path) of a file where workflow definitions are stored after exporting from a 3.x system, and before importing to a 4.4.x system. This is required when the mode is "export' or "import". This is optional when the mode is "both". Example: `-f intermediate.txt`
-v	`--validate-only`	Optional. When specified, this prevents anything from being written to the 4.4.x system. It can be used with any of the three modes, but has an effect when the mode is "import", or "both".

-d	`--database`	Connection string (in JDBC format) for the database containing workflow definitions (and associated data) to be migrated.
		Either a database connection string or an input file name must be specified if the mode is "export" or "both".
		Example:
		`-d "jdbc:sqlserver://<server>;`
		`databaseName=<database>;user=<user>;password=<password>"`
		and
		`-d "jdbc:oracle:thin:<user>/<password>@<server>:<port>:<sid>"`

-i	`--input-file`	Name of a single file or directory containing exported OpenEDMS definitions. If specified, the file or files must be in valid XML format, as exported by the OpenEDMS Designer tool. If the supplied name is a directory, the migration tool will attempt to process files in that directory. **Note:** The OpenEDMS designer exports two files for each workflow definition. One file has no extension but contains the workflow definition XML.

Flag	Long form	Description
		The other file has an ".xml" extension and contains the UI design information for the workflow. These "design" files are ignored by the migration tool. Only the workflow definition files are processed. Example: `-i myFile` `-i c:\Data\OpenEDMS_Workflows`
-s	`--target-server`	Base URL of the 4.4.x server where the definitions will be imported. This is required if the mode is "import" or "both". This is optional if the mode is "export". If used with "export" mode, the target server will be used for data validation purposes only. If the target server is specified, the following additional options must also be specified: -a (`--authentication-server`) -u (`--user-id`) -p (`--password`) The target server URL must include the protocol (HTTP or HTTPS), and the port (typically 8080). Example: `-s http://myServer:8080`
-a	`--authentication-server`	The URL of the OpenAM server through which the supplied user credentials will be validated. The URL must be of the form shown in the example below. This is required if the target server (-s or `--target-server`) is specified. Example: `-a http://myOpenAmServer:8080/OpenAM`
-u	`--user-id`	ID of the account used to access target server web services. This account needs to have permission to read principals, domains, folders and templates and to create workflow definitions and associations.

Flag	Long form	Description
		The sysadmin account is typically used. The user ID is required if the target server (-s or `--target-server`) is specified. Example: `-u sysadmin`
-p	`--password`	Password corresponding to the user ID specified through the –u or `–user-id` flag. The password is required if the target server (-s or `--target-server`) is specified. Example: `-p myPassword`
-c	`--community`	Name of a "domain" (EngageOne 3.x versions), or "community" (EngageOne 4.4.x versions). This flag is optional. When 3.x definitions are read from a database: • If the community name is provided, the workflow definitions in a domain with the specified name will be exported. These definitions will be imported into a community with the same name. • If the community name is not provided, the migration tool will process workflows in all domains contained in the database, and will import definitions into the named communities in the target system. When 3.x definitions are read from files: • If the community name is provided, this will override the domain ID in the XML, and the workflows will be added to that community in the target system. If no community with the specified name is in the target system, the workflows will not be imported. • If the community name is not provided, the domain ID in the XML definition will be used to look up the community in the target system. If there is no community with a matching ID, the workflows will not be imported.

Flag	Long form	Description
		Note: When reading 3.x definitions from files, the –c flag can be used with any of the three modes (export, import, both). If a two-phase approach is used (export to an intermediate file, followed by an import to the target system) the –c flag should be used with the export, the import, but not used with both. Example: `-c myCommunity`
-y	`--active-only`	Optional (the default is False). If specified, the migration tool will only export or import workflow definitions marked as "Active" in the source system. Note that only current workflow definitions are considered for migration and earlier versions, "Active" or "Inactive", are ignored. If the current version is inactive, no version will be migrated. This applies if an earlier version is active. **Note:** In EngageOne 4.4.x, workflow definitions are always active. If non-active definitions are read from the 3.x system (when the –y flag is not set), those definitions will become active when imported to the 4.4.x system. Example: -y
-t	`--associate-template`	Optional (the default is False). If specified, the migration tool will attempt to associate the migrated definitions with document templates in the target environment that correspond to the associated templates in the source system. The tool matches templates based on their ID, not their name. A template must have the same ID in both the source and target systems. This is the case if the EngageOne database has been migrated from 3.x to 4.4.x. If the OpenEDMS definitions are read from XML files the –t flag will have no effect, the XML files do not contain any information about template associations. Example: -t

Flag	Long form	Description
-l	`--associate-folder`	Optional (default is False). If specified, the migration tool will attempt to associate the migrated definitions with folders of document templates in the target environment that correspond to folders in the source system. The tool matches the folders by path name. **Note:** In EngageOne 3.x, it is possible to associate workflows with folders. This capability is not fully functional and the folder associations do not get taken into account when the a workflow process is started. Care should be taken when using this flag as the system may behave differently if (fully functional) folder associations are created in the 4.4.x system. If OpenEDMS definitions are read from XML files the –l flag will have no effect, because the XML files do not contain information about folder associations. Example: -l
-n	`--create-instance`	Optional (default is False). If specified, the migration tool will identify in-flight work items, correlate them with migrated workflow definitions and start instances in the target environment for each one. The -n flag can only be used when the tool is run in "both" mode and also requires values for `--database` and `--target-server` flags. **Note:** When the tool is run in "export" mode, the number of in-flight work items will be read from the source system database and reported as an indication of how many records might be migrated. Example: -n

Reference information

Limitations

OpenEDMS workflow uses a different conceptual model than the new workflow system.

The migration tool applies a set of rules to ensure that each workflow is compatible with the new system. If any incompatibilities are identified the workflow will not be processed.

The rules are:

- Workflow must have exactly one submission (start) step, and this step must have at least one participant.
- Workflow must have exactly one decision (assignment) step, and this step must have at least one participant.
- The decision step must not have any routes defined.

By examining some examples of how customers use OpenEDMS workflow, we believe the above rules will have no impact.

In any cases where these rules prevent workflows from being migrated, it will be necessary to identify and manually implement a workaround.

Submitters and approvers

In OpenEDMS workflow each step (submission and decision) has participants assigned to it.

For a submission step, the participants represent users who have submitted the communication.

The meaning of participants in this context is "Only create a workflow of this type if the document was submitted by one of these participants". In the 4.4.x workflow system, the equivalent functionality is provided by the "applicable principals" list on a workflow.

For a decision step, participants represent users who need to review or approve the communication. In the 4.4.x workflow system, the equivalent functionality is the list of reviewers for a review step in a workflow.

OpenEDMS has the concept of a "route" on the submission step.

This is a mapping from submitters to reviewers.

For example, a route from "A" to "B" means that when a communication is submitted by participant "A" it must be reviewed by participant "B". Multiple routes can be configured for each submitter and each reviewer can be "routed to" from multiple submitters.

The concept of routing does not exist in the 4.4.x workflow system. The same functionality can be approximated by having multiple (similar) workflow definitions, each with a different set of "associated principals" (submitters) and reviewers.

Each 3.x workflow definition might be represented by multiple workflow definitions in 4.4.x. If multiple workflows are created from a single source workflow their names have a number added at the end to make them unique.

For example, a 3.x "MyWorkflow" might be represented by "MyWorkflow_1", "MyWorkflow_2" and "MyWorkflow_3", in 4.4.x.

In-flight work Items

In-flight work items are migrated at the same time as their corresponding workflow definition.

If the workflow definition exists in the target system when the tool is run, corresponding work items will be ignored.

Note that only current versions of workflow definitions can be migrated. Work items created from earlier versions will be ignored.

When a work item is migrated, a new instance of the corresponding workflow definition will be started in the 4.4.x workflow system. This means a new approval process, so work items in a SUBMITTED_REJECTED status will be updated to a PENDING_APPROVAL status.

Streamline upgrade

It is important to note that this upgrade method can only be performed to the latest 3.1.2 and later versions of EngageOne Server.

The system upgrade consists of multiple predefined steps, some of which must be performed in a specific order. Many of these steps are carried out automatically, while others require manual intervention.

You can perform the manual steps in a separate console while the automatic steps are running. Alternatively, you can interrupt the upgrade and perform the manual steps; when the upgrade restarted, it will resume processing at the point where it stopped.

Preparation

Before starting the migration process, backup the EngageOne server database (SQLServer or Oracle), the active-drive, and the EngageOne server installation folder.

Depending on your database infrastructure, and the computer where the upgrade will be performed, the following programs must be installed:

- MsSQL:
  - sqlcmd Utility from Microsoft, refer to **https://docs.microsoft.com/en-us/sql/tools/sqlcmd-utility?view=sql-server-ver15**
- Oracle:
  - Instant Client Base
  - Instant Client SqlPlus

  Refer to **https://www.oracle.com/database/technologies/instant-client.html** for details
- Groovy, minimum version level 2.5.6.

On the migration server (primary Core bundle is recommended), ensure that you have followed the sections **Extracting the release distribution** on page 5 and **Installing scripting prerequisites** on page 9

Extract the release distribution to a temporary location, referred to as <release-distribution> below; the folder structure is as follows:

```
<release-distribution>/
    /bundles
```

```
    /docs
    /install
    /samples
    /upgrade
      migrate.groovy
      streamlineUpgrade.groovy
      /active-drive
      /database
      /mssql
      /oracle
      /migration
             /communities-migrator
             /template-indexer
  /workflow-engine-migrator
  /workflow-migrator
       /workitem-migrator
    /utilities
```

Points to note

- The upgrade takes care of all database update scripts and runs all relevant migrations.
- The upgrade informs you when the software installation is required. This must be performed manually by the system administrator.
- Before starting the migration process, you must prepare the appropriate deploy.properties files for EngageOne Compose v4.4.10.
- Reuse the existing active drive and do not create a new one.
- When migrating from version 4.x.x and later, it is recommended to copy existing files which then need to be modified to meet the requirements of v.4.4.10, as described in **Deployment configuration upgrade** on page 113
- When migrating from the v3.1.2:
  - you must prepare a new deploy.properties file as described in **Establishing system configuration** on page 12.
  - a flowable database must be created by the systems administrator as detailed in **Databases** on page 46.

activiti to flowable deploy.properties update

You are required to change activiti references to flowable in your deploy.properties file as follows. Note that this change is relevant to upgrades from 4.3 and later versions of EngageOne Server.

From:

```
activiti.db.name=
activiti.db.username=
activiti.db.password=
```

To:

```
flowable.db.name=
<when migrating value should be the same as it was for
activiti.db.name>
flowable.db.username=
<when migrating value should be the same as it was for
activiti.db.username>
flowable.db.password=
<when migrating value should be the same as it was for
activiti.db.password>
```

SQL considerations

To perform certain actions the process must execute either `SqlCmd` or `SqlPlus`, as mentioned above.

By default, these are:

Microsoft SQL Server

```
c:\Program Files\Microsoft SQL Server\Client
SDK\ODBC\110\Tools\Binn\sqlcmd.exe
```

Oracle

```
c:\oraclexe\app\oracle\product\11.2.0\server\bin\sqlplus.exe
```

Add an `install.sql.cli` parameter to your `deploy.properties` file, to indicate the path and executable should you choose not to use the default settings.

For example:

```
install.sql.cli=C:/Program Files/Microsoft SQL Server/Client
SDK/ODBC/170/Tools/Binn/SQLCMD.EXE
```

Performing the update

The *streamlineUpgrate.groovy* script is located under *<release distribution>/upgrade*.

The script is run by issuing the following command:

```
groovy streamlineUpgrade.groovy -p deploy.properties <from_version>
```

You must specify the currently installed version of EngageOne Compose to be upgraded in the *<from_version>* parameter. The update script provides information about the current status and required manual actions when needed.

Note if the script is be aborted for any reason, a record is kept as to the stage at which the process was aborted. When the script is restarted, it will resume exactly where it left off and continue the update process.

Restarting an update from the beginning

If for any reason you need to start the upgrade process from the beginning, you must run the *streamlineUpgrade.groovy* script using an additional reset parameter as shown below:

```
groovy streamlineUpgrade.groovy -p deploy.properties reset <from_version>
```

It is recommended to restore a backup of the database, active-content, and directory with the current version of the application before restarting the update. Performing an upgrade on already partially upgraded structures may lead to data corruption.

15 - Troubleshooting Installation

In this section

Script configuring the Security bundle never completes

The script that configures the Security bundle may not complete if the following conditions exist:

- You are installing on Windows, and
- The file path to the Security bundle install location contains spaces, and
- 8dot3 file name creation is disabled on the server's file system.

Note:

To find out the 8dot3name setting on your file system run the command `` `fsutil 8dot3name query c:` ``

To work around this issue:

- Change `security.install.dir` to a directory that *does not* contain spaces. Or,
- Enable 8dot3 file name creation using the command `` `fsutil behavior set disable8dot3 0` ``

Troubleshooting Windows SSO

If you experience problems in enabling single sign-on to EngageOne applications, refer to the following resources to help you troubleshoot:

- **http://troubleshootingrange.blogspot.co.uk/2012/08/kerberos-desktop-sso-with-openam-and.html**
- **https://backstage.forgerock.com/knowledge/kb/article/a14556843**
- **https://docs.oracle.com/cd/E19575-01/820-3746/gisep/index.html**

Troubleshooting SSL connection issues

The sslPoke.groovy script is provided for SSL troubleshooting issues. This script can be found in install folder of the EngageOne distribution zip file.

The script can be used for investigating / troubleshooting SSL connections to various parties, by using the SSL properties (truststore, ciphers, protocol version) specified in your deploy.properties file.

usage:

```
groovy sslPoke.groovy [options...]
```

Where [options...] can be replaced by:

- `--debug` or shortened `-d`

  sets `-javax.net.debug=all`

- `--host` <arg> or shortened `-h` <arg>

  where: <arg> is the Host address and port that ssl connection will be checked on. For example, `my-very-secured-ldap.domain.net:636`

- `--properties` <arg> or shorten `-p` <arg>

  where: <arg> is the deploy.properties path

16 - Appendix

In this section

System Services

The five Web application bundles (Security, Core, Composition, Conversion, and Notification) run as system level services on the target operating system. For Windows Server, this is a Windows Service and on Linux a SysVinit script is created in the `/etc/init.d` folder. The `configure` target of `eos.groovy` will uninstall and install the service on each invocation so that the relevant properties in `deploy.properties` are applied to the service.

Windows Service

The services run under the following names:

- EngageOne Server Composition Service
- EngageOne Server Conversion Service
- EngageOne Server Core Service
- EngageOne Server Security Service
- EngageOne Server Notification Service

The service is installed with a 'Startup Type' of Automatic and will start at system boot time.

Start and Stop Services

The services are started and stopped from the Service application on your Windows Server.

1. Open the run dialog from the Windows start button.
2. Type `services.msc` in the search box.
3. Click **Enter**.
4. Locate the service by name (see above) and click **Start**, **Stop** or **Restart** on the left panel.

Service user

By default, the services will run under the 'Local System' account ('Log On As' setting in the Services application). To modify this user, set a username and password in the following in `deploy.properties`:

- `os.service.username`
- `os.service.password`

This account must have the 'Log on as a service' privilege and have read/write access to the `${bundle.install.dir}` and the `${active.drive.dir}` folders.

Linux Service

In a Linux environment the applications are started using the SysVinit style initialization. The `eos.groovy` script creates a startup script under `/etc/init.d` for each bundle. These scripts are named as follows:

- `eos-composition`
- `eos-conversion`
- `eos-core`
- `eos-security`
- `eos-notification`

Start and Stop Services

Use the Linux service command to stop and start the services:

```
service eos-core start
service eos-core stop
```

Service user

By default, the service is configured to run as the 'root' user. To modify this user set the following in `deploy.properties`:

- `os.service.username`

For Linux the password is not necessary. This account must have read/write access to the `${bundle.install.dir}` and the `${active.drive.dir}` folders.

Creating PKCS#12 Archives with Java keytool

The TLS certificates used to secure the services may be obtained from a certificate vendor or created internally depending on client preference and policies. The Java keytool is one of a number of utilities that can be used to both create new self-signed certificates and add existing/externally sourced certificates.

In some cases it is sufficient to create a single certificate, for example:

```
keytool -genkey -keyalg RSA -keystore eokeystore.pfx -storepass
My_Pa55w0rd -validity 1024 -keysize 2048 -deststoretype pkcs12 -dname
"CN=*.mydomain.int.pvt"
```

If you wish to protect each server with its own certificate you will be required to combine all of the certificates into a single archive and give each a unique alias. For example:

```
keytool -genkey -keyalg RSA -alias core -keystore eokeystore.pfx
-storepass My_Pa55w0rd -validity 1024 -keysize 2048 -deststoretype pkcs12
 -dname "CN=eos.mydomain.int.pvt"
keytool -genkey -keyalg RSA -alias cservice -keystore eokeystore.pfx
-storepass My_Pa55w0rd -validity 1024 -keysize 2048 -deststoretype pkcs12
 -dname "CN=eocs.mydomain.int.pvt"
keytool -genkey -keyalg RSA -alias designer -keystore eokeystore.pfx
-storepass My_Pa55w0rd -validity 1024 -keysize 2048 -deststoretype pkcs12
 -dname "CN=eodesigner.mydomain.int.pvt"
```

The example above works for a simple non-clustered installation but to support multiple nodes you must create a SAN (Subject Alternate Name) extension for each node. Expanding on the example above, add two nodes to each certificate:

```
keytool -genkey -keyalg RSA -alias core -keystore eokeystore.pfx
-storepass My_Pa55w0rd -validity 1024 -keysize 2048 -deststoretype pkcs12
 -dname "CN=EngageOne Core Services" -ext
SAN=dns:eosn1.mydomain.int.pvt,dns:eosn2.mydomain.int.pvt
keytool -genkey -keyalg RSA -alias cservice -keystore eokeystore.pfx
-storepass My_Pa55w0rd -validity 1024 -keysize 2048 -deststoretype pkcs12
 -dname "CN=EngageOne Conversion Services" -ext
SAN=dns:eocsn1.mydomain.int.pvt,dns:eocsn2.mydomain.int.pvt
keytool -genkey -keyalg RSA -alias designer -keystore eokeystore.pfx
-storepass My_Pa55w0rd -validity 1024 -keysize 2048 -deststoretype pkcs12
 -dname "CN=eodesigner.mydomain.int.pvt"
```

Note: The Designer services are single node, no SAN extension is necessary.

Server status and health check endpoint

Each bundle exposes a status or health check endpoint you may use to check system availability. If the server is up and running, an HTTP `GET` on the URL will return a status code of 200. The URL uses the resource `status` at the root of each bundle.

For example - `http://core.node1:8080/status`. This URL may be used in your load balancer configurations (to determine server availability) and system monitoring tools.

In addition to the 200 success response code, the endpoint displays a simple page of status information or verification outputs. Those outputs include:

- Server Description - a description which includes the type of bundle running (Core, Security, etc.)

- HTTP Request Headers - a dump of the HTTP request headers seen by the application server. This is often helpful in troubleshooting load balancer configurations and verifying the X-Forwarded-XXX headers.
- Application Versions - displays the set of Web applications and their version numbers deployed.

Working with the patcher script

The patcher is a groovy script distributed with the installation media to automate the process of implementing patches to EngageOne Server software. Patches are issued by the support team to resolve urgent issues that may be encountered in your environment.

Definition of terms

Term	Definition
Active Drive	The Active Drive, or Shared File System is an external component used by EngageOne Server to store persistent and temporary files outside of the SQL database.
deploy.properties	deploy.properties is the configuration file specified at installation time to configure all bundles. It must be populated before installation time with configurations such as database connection information, DB connection information and the location of the active-drive.
Groovy	Groovy is a scripting language that runs within the Java Runtime. EngageOne Server's installation scripts are written in Groovy.
bundle	EngageOne Server is deployed as groups of applications known as bundles. There are six bundles – Security, Core, Composition, Conversion, Notification and Batch

Prerequisites

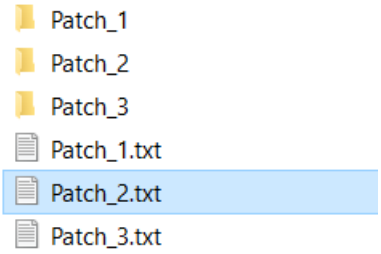
- EngageOne SP10 or above
- Java (refer to, **Installing Java** on page 9)
- Groovy(refer to, **Installing Groovy** on page 10)
- Permissions to read/write to shared Active Drive location
- Permissions to read/write to /etc in Linux, or `%SYSTEM_DRIVE%/ProgramData` in Windows
- The machine on which the patcher script is to be executed must have access to the database

About the patching process

- The patcher script is controlled by a text file which describes the operations to be performed by the patcher script. This file is provided by the support team and must be copied to the following location on your active-drive:

```
...\active-drive\patcher
```

- The patcher can run in two modes, either to perform the patching process or to rollback from the previous patching process. Refer to
- A patch consists of a single tcxt file and a corresponding folder containing the files required for the patch, as shown below:



You may be supplied with a number of dependent patches that must be run in a particular order. In this scenario, the patch process will manage dependencies, however, you must run the last patch in the dependency first. In the example above, if the patch order is:

Patch_3 –depends on–> Patch_2 –depends on–> Patch_1

you must run the patch script with Patch_3 as the -c switch parameter on the command line.

An example patch context file

In the example patch context file that follows:

- The core section is applicable to the core bundle and instructs the patcher to:
  - patch project-services.war.
  - add first.txt and removes second.txt.
- The security section is applicable to the security section and instructs the patcher to add third.txt.
- The sqlScriptsToExecute section to run the mysql script.

```
{
 "id": "sample_patch_idtest",
 "version": "4.4.10",
 "casesFixed": ["ces-4324234"],
 "bundlesPatched": {
  "core": {
   "patchedFiles": {
    "deployments": [{
     "file": "project-services.war"
    }]
   },
   "addedFiles": {
    "bin": [{
     "file": "First.txt"
    }]
   },
   "removedFiles": {
    "bin": [{
     "file": "second.txt"
    }]
   }
  },
  "security": {
   "addedFiles": {
    "bin": [{
     "file": "third.txt"
    }]
   }
  }
 },
 "sqlScriptsToExecute": {
  "MSSQL": ["mssql.sql"]
 },
}
```

Running the script

Your may have multiple EngageOne Server environments, each may have varying bundles installed; the patcher can be run in all environments. The patcher script will filter out the bundles that are not installed, and will patch only those that are present in the environment.

Below is the usage statement of this script:

```
usage: groovy patcher.groovy [options] [target] [target2] [target3]...
-c, --context, <arg> Specifies the location of context file for current
 patch
-h, --help Usage information.
-p, --properties, <arg> Specifies the path of the deploy.properties file
 containing EngageOne Server
    onfiguration information.

targets:
patch        -> Patch a target bundle.
rollback     -> Rollbacks previously made patch
```

The patching process ensures all modified files are backed up to the Active Drive shared location and an entry is created in the `SYSTEM_CONFIGURATION` database table. When rollback is required, all files will be restored from previously backed up version from Active Drive shared location, at which point the database entry will be deleted.

Before the patching process takes place, a backup of all changed files is made. The backup is stored at the following location:

```
Active Drive/patcher/<patch_id>/backup/<node_name>/<node_UUID>
```

Rollback

The rollback process takes into account patch dependencies. For example, if the patch dependency is:

Patch3 –depends on–> Patch2 –depends on–> Patch1

When the patcher is run with rollback using Patch1, the patcher will rollback in the following order:

Patch3 –> Patch2 –> Patch1

Note that the rollback uses files that were previously copied to backup folder. After successful rollback, information about patch is deleted from database and also from backup folder.

Migration script

An additional script, `migrate.groovy` is found in the `<release-distribution>/upgrade` folder and may be used to migrate existing data. It is a target based interface allowing you to perform different operations with different invocations.

Below is the usage statement of this script:

```
usage: groovy migrate.groovy [options] [target [target2 [target3]...]]
 -h,--help                Usage information.
 -l,--logback <arg>      Specify the location of the logback configuration

                          file.
 -p,--properties <arg>   Specify the location of the properties file
                          containing configuration information.
targets:
 active-drive    -> Update active drive structure.
 community       -> Migrate the viewpoint communities.
 template        -> Migrate the EngageOne templates.
 workflow        -> Migrate workflow definitions and instances from EO
3.2.1 to 4.4.10+
 workflow-engine -> Migrate workflow definitions and instances from EO
4.x to 4.4.10+
 workitem        -> Migrate work items.
```

The properties file shares a subset of the configuration data as documented for `deploy.properties` in the *EngageOne Server 4.4: Installation Guide* and must adhere to all the requirements for the new software installation. The properties file will be referred to as `deploy.properties` for the remainder of this document.

By default, migration tools run with logging level set to WARN. To change this, specify the location of a logback xml configuration file and increase the console logging level. For example:

```
<?xml version="1.0" encoding="UTF-8"?><configuration>
 <shutdownHook class="ch.qos.logback.core.hook.DelayingShutdownHook"/>
 <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
   <level>INFO</level>
  </filter>
  <encoder>
   <pattern>[%date | %-5level | %1logger] %msg%n</pattern>
  </encoder>
 </appender>
 <root level="INFO">
  <appender-ref ref="console"/>
 </root>
</configuration>
```

Advanced install script topics

Configuring logging

All of the install scripts use logback to log to standard out (shell or console window) and the log files. The configuration can be found in `<release distribution>\install\groovy\lib\resources\logback.xml`. By default, the scripts log in 'INFO' mode to both destinations with some additional details (fatal error stack traces) to the log file. In the event you need additional logging information or would like to redirect it to a different location you may edit this configuration. A typical scenario is the need to increase the logging data this may be accomplished by setting the root logger level to `DEBUG`:

```
<root level="DEBUG">
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="SIFT"/>
</root>
```

Logback supports a number of advanced features which are documented on their website - http://logback.qos.ch/.

Special software requirements for Linux

If you are installing EngageOne Server on a Linux server, the following requirements must be met.

Installing the libpng library

Use the appropriate package manager (for your Linux distribution) to identify and install the most recent libpng artifacts on the Conversion bundle.

1. To identify the right version execute the `whatprovides` command for your distribution:

Note: The following uses "libpng12" as the artifact for this example. Substitute the version number that is returned when executing the `whatprovides` command.

```
yum whatprovides libpng12.so.0
```

or

```
zypper search --provides libpng12.so.0
```

2. Install the version identified.

```
yum install libpng12-1.2.50-6.el7.i686
```

or

```
zypper install libpng12-0
```

Installing libssl3 library (SUSE only)

Use the Linux 'zypper' utility to install a 64-bit libssl3 artifact. This is a library within the Network Security Services (NSS) package. This is required for the Conversion and Composition bundles. An install example is shown for the mozilla-nss library.

```
> zypper addrepo
http://download.opensuse.org/repositories/home:draht/SLE_11_SP3_Update/home:draht.repo
> zypper refresh
> zypper install mozilla-nss
```

64 bit Linux considerations

Certain 32bit libraries are required which may not be supplied with recent 64 bit systems. To ensure the required libraries are available, follow the steps described below.

add setting: multilib_policy=all to yum.conf

```
yum install libstdc++*
yum install zlib*
yum install libncurses*
```

Note: yum commands are presented above, use the corresponding commands for other packet managers.

Note: If any of the commands fail, retry the appropriate command using the following command option:

```
--skip-broken
```

Adjusting limits in Linux

Some Linux distributions have default limits on concurrent threads run by a user (or system).

The EngageOne Server Composition bundle may require that thread limits be adjusted.

To check the current limit run:

- `ulimit -u`

To create extra process capacity on your system, edit the `limits.conf` file, located in `/etc/security/limits.conf`.

- Add or modify the lines for `nproc`.

For example,

- `engageOneProcessUser soft nproc 4096`
- `engageOneProcessUser hard nproc 4096`

Where "`engageOneProcessUser`" is the Linux user running the EngageOne Server service.

Note: Soft and hard limits need to be adjusted.

Running the installation as a non-root privileged Linux user

You must be a non-root privileged user when installing EngageOne on a Linux server. In this section, the steps required to create a non-root user account on Linux environment that will run the EngageOne Server suite installation are provided and should be followed in conjunction with the information described in **Installing and configuring bundles** on page 52.

Non-root privileged users have reduced scope to perform standard operations, but will also have privileged access when needed. You must follow the information in this section in the order presented.

Create EngageOne group

The engageone group must be created either by a `root` or privileged user. Firstly, a new ***engageone*** group that the user will be assigned to must be created:

```
# groupadd engageone
```

Create EngageOne user

A new `engageone` user must now be created. This user will become a member of `engageone` group, with a:

```
/opt/engageone
```

as a home directory; this is where EngageOne Server will be installed:

```
# useradd -m -g engageone -b /opt engageone
```

Specify `engageone` user password by issuing and typing password:

```
# passwd engageone
```

Add EngageOne user permissions

This section describes the system/file privileges/permissions an `engageone` user must have in order to administer and run the EngageOne Server suite.

Home directory permissions

Make the `engageone` user and its group owner of its home directory:

```
# chown -R engageone:engageone /opt/engageone
```

Root privileges

In order to perform administrative tasks related to the installation, maintenance and running of the EngageOne Server suite, the `engageone` user must be granted root privileges. This allows the non-root `engageone` user to run commands with administrative privileges by using `sudo` for

commands that require it. There are a number of ways to accomplish that. Here, we run through two way which may be used;

by adding the:

- `engageone` user to the `sudo` group
- `engageone` group to `sudoers`

Add engageone user to sudo group

There are general-purpose administration groups on Linux system if such group exists then `engageone` can be assigned to this group to obtain administrative privileges and execute commands using `sudo`.

On CentOS/RHEL/SUSE there is a ***wheel*** group that provides privileged access. It is important to ensure the ***wheel*** group is uncommented in the `/etc/sudoers` file, as follows:

```
# gpasswd -a engageone wheel
```

It may be required to log in again as the `engageone` user for this change to take an effect.

Add EngageOne group to sudoers

You will be required to edit the `/etc/sudoers.d/engageone` file with appropriate entry as follow

Edit the `/etc/sudoers.d/engageone` file with command:

```
# visudo -f /etc/sudoers.d/engageone
```

Next, add following line to allow all commands to be executed as root by `engageone` group member:

```
## Allow users in engageone group to run all commands as root
 %engageone ALL=(ALL) ALL
```

System administrators can limit command access based on commands that the `engageone` user must be able to run using `sudo` during installation and maintenance. For details on what commands are used during EngageOne Server installation and maintenance, please refer to **Linux commands used** section.

A more restrictive example of */etc/sudoers.d/engageone* file can be as follows:

```
## Command aliases for engageone group
## Installation commands
Cmnd_Alias EO_INSTALL = /bin/chmod, /bin/chown
## Maintenance commands
Cmnd_Alias EO_MAINTENANCE = /sbin/service, /sbin/chkconfig, /sbin/runuser,
 /bin/bash, /bin/su, /bin/sudo -s
## Run as aliases for engageone
Runas_Alias ADMIN_ALIAS = root
## Allow users in engageone group to run specified commands during
```

```
installation as root
%engageone ALL=(ADMIN_ALIAS) NOPASSWD: EO_INSTALL, EO_MAINTENANCE
## Allow users in engageone group to run specified commands during
maintenance as root
#%engageone ALL=(ADMIN_ALIAS) NOPASSWD: EO_MAINTENANCE
```

When a dedicated user from the `engageone` group performs an installation then the section including both `EO_INSTALL` and `EO_MAINTENANCE` command aliases must be active (uncommented).

After installation, when a user from the `engageone` group performs maintenance activities, then its root privileges can be further limited by activating (uncomment) the section with only `EO_MAINTENANCE` command alias (and commenting out section containing `EO_INSTALL` and `EO_MAINTENANCE` command aliases).

System resource limits adjustments

There is a need to adjust maximum number of processes a user can run on the system.

For root users, NPROC usually is set as a large number whereas, this value for non-root user it is very limited. There are various process-intensive operations within the suite especially in the Composition bundle; when this setting is set to a low there is a possibility that errors may be visible in the logs such as:

```
java.lang.OutOfMemoryError: unable to create new native thread
```

To adjust the NPROC, NOFILE for the `engageone` user, create an engageone.conf file in: `/etc/security/limits.d`: as follows:

```
# vi /etc/security/limits.d/engageone.conf
```

Add following lines to the `engageone.conf` file:

```
# Custom limits for engageone user.
engageone soft nproc 4096
engageone hard nproc 4096
engageone soft nofile 10000
engageone hard nofile 10000
```

IMPORTANT: note that recommended settings specified here are set per **single USER ID** (***engageone)*** and for single EngageOne Server bundle run by this user on the server. Which means that those limits are **shared** between EngageOne Server bundles when run on the same machine using same ***engageone*** user. In this case, those limits **must** increase according to the number of bundles run using same user on the same server.

As limits here are set by *pam* when login session starts you must ensure that the following entry is present in the `/etc/pam.d/system-auth-ac` file:

```
session required pam_limits.so
```

Using *.conf file placed in `/etc/security/limits.d` directory enforces resource limits for the users logged in via `PAM`. For this change to be applied it is necessary for the `engageone` user must log-out and log back in again.

It is also important note that all files present in the: `/etc/security/limits.d` directory are processed in an alphabetical order. All limits from the file read later will **override** any previous settings. It is therefore necessary to ensure that none of the files loaded after `engageone.conf` file will override the NPROC or NOFILE limits, by specifying a lower number for all users using * (the wildcard sign).

EngageOne Server processes a large number of files when in operation. It is essential that the user running EngageOne Server services have the appropriate limit set for the number of open files. It is not possible to set an unlimited value. A seeting of 10000 concurrent file descriptors, which can be open, by a single process is recommended. This value can be adjusted by system administrators according to their knowledge as well as the number of concurrently open file descriptors allowed by the kernel, which can be checked in `/proc/sys/fs/file-max` file. If required */proc/sys/fs/file-max* may also be adjusted and increased where the server specification allows; in this scenario it is important to be aware of the `/proc/sys/fs/nr_open`, which has an upper limit of `fs.file-max`.

EngageOne Server installation modifications

Certain changes are required to the standard installation setup for a non-root user to be used.

Creating installation directory

The EngageOne Server release distribution is provided in a single *zip* file. The release distribution should be copied and extracted to a temporary location on each application server node using created `engageone` user. If for some reason, a different user was used to extract release distribution please change ownership of this directory to `engageone:engageone` by issuing following command:

```
# chown -R engageone:engageone /opt/engageone/tmp/release_dir
```

Deploy.properties adjustments

In order to instruct our System V init scripts to run EngageOne Server bundles as a specific non-root user the `os.service.username` property in the `deploy.properties` configuration file must be set with `engageone` user name, as follows:

```
os.service.username=engageone
```

EngageOne Server installation procedure

In this section, we will present how to install our bundles as the non-root privileged user that was previously set up.

You must log in as the `engageone` user, all steps in this section must be executed by this user.

Refer to **Preparing for installation** on page 4 for information on how to prepare environments, install all necessary tools and software.

Install bundle

First, execute the `install` target to copy the specified bundle to the target install location:

```
$ groovy eos.groovy -b <bundle> -p <deploy.properties path> install
```

Validate configuration (Optional)

This optional step can be executed in order to validate `deploy.properties` settings for bundle being installed. Run `validate` target, as follows:

```
$ groovy eos.groovy -b <bundle> -p <deploy.properties path> validate
```

The validation performs checks of properties specific for bundle being installed and will fail with a warning and guidance on what actions to take if any configuration properties have not been specified correctly.

Configure Bundle

This final step in the install process is to run `configure` target. Because EngageOne Server scripts use System V init mode, it is necessary to elevate privileges of the `engageone` user in order for the `configure` target to be executed without permission errors.

As a result of the `os.service.username=engageone` modification applied to the `deploy.properties`, all EngageOne Server services will be run by the `engageone` user. Depending on your installation, you must specify appropriate node type (-t) value. A standalone installation this setting is as follows:

```
$ sudo -s
# groovy eos.groovy -b <bundle> -p <deploy.properties path> -t single
configure
```

IMPORTANT: Once the bundle `configure` has completed, certain directories/files are created with `root` user as the owner. This needs to be changed and ownership of those directories/files needs

to be re-assigned back to the `engageone` user and group; this is accomplished using the following command:

```
# chown -R engageone:engageone /opt/engageone/server/<bundle directory>
```

Ownership of all the files and directories under specific bundle directory is reverted back to the `engageone` user and its group.

EngageOne Server maintenance commands

In this section, we outline some of the most common operating system commands used for EngageOne Server maintenance purpose. All commands are executed by `engageone` user.

Stop bundle's service

```
$ service eos-core stop
```

Start bundle's service

```
$ sudo service eos-core start
```

Run configure target for specific bundle

```
sudo -s
# groovy eos.groovy -b <bundle> -p <deploy.properties path> -t single
configure
# chown -R engageone:engageone /opt/engageone/server/eos/<bundle>
# exit
```

IMPORTANT: After bundle configuration, certain directories/files are created with `root` user as owner. Ownership must be changed and re-assigned back to the `engageone` user and group. After successful installation of particular bundle the `chown` command must be executed.

Uninstall bundle's service

```
$ sudo -s
# groovy uninstall.groovy -b <bundle> -p <deploy.properties path> service
# exit
```

Uninstall bundle

```
# groovy uninstall.groovy -b <bundle> -p <deploy.properties path> bundle
```

EngageOne Server Linux commands in use

This section details all Linux commands, actions and file system items that non-root user must have privileges to use/execute/access in order to accomplish EngageOne Server installation and maintenance tasks.

Linux commands used

Command	Requires SUDO for Installation	Requires SUDO for Maintenance	Comments
/sbin/service	Yes	Yes(*)	service command being used to operate System V init script. (*) It is required to use SUDO for service start all other operations should be available without it
/bin/ps	No	No	ps used to list current processes
/bin/chmod	Yes	N/A	chmod command is used to change created/copied files mode
/sbin/chkconfig	Yes	Yes	chkconfig command is used to update runlevel for system services.
/bin/chown	Yes	No	chown command is changing file owner and group

Command	Requires SUDO for Installation	Requires SUDO for Maintenance	Comments
/sbin/runuser	Yes	Yes	runuser command is used to run EngageOne Server bundles with dedicated user and group ID
/bin/su	Yes	Yes	su command is used to run EngageOne Server bundles with dedicated user and group ID
/bin/kill	No	No	kill command is used to terminate EngageOne Server bundle process

File system resources assess

- All bundle installation directories (and all its subdirectories) must be owned by non-root user created and its group (***engageone:engageone***).
- User used must have read/write access to the Active Drive specified in deploy.properties configuration file.
- /etc/init.d/ - System V init scripts are in this directory. Either Non-root user must have write/read access to this location or, the installation script would have to be run with elevated privileges (root).

Enabling LDAP over SSL/TLS

This section covers the steps you must follow to enable LDAP over SSL/TLS (or LDAPS), in the Microsoft Active directory of Windows Server 2012 R2.

LDAPS certificates in Active Directory

Microsoft defined a strict criteria about the certificates used for Active Directory to enable LDAP over SSL/TLS (**https://support.microsoft.com/en-us/kb/321051**).

There is no user interface for configuring LDAPS. Installing a valid certificate on a domain controller permits the LDAP service to listen for, and automatically accept, SSL connections for both LDAP and global catalog traffic.

Requirements for LDAPS Active Directory server certificate

To enable LDAPS, you must install a certificate on the LDAP server that meets the following requirements:

- The LDAPS certificate is located in the Local Computer's Personal certificate store (programmatically known as the computer's MY certificate store).
- A private key that matches the certificate is present in the Local Computer's store and is correctly associated with the certificate. The private key must not have strong private key protection enabled.
- The Enhanced Key Usage extension includes the Server Authentication (1.3.6.1.5.5.7.3.1) object identifier (also known as OID).
- The Active Directory fully qualified domain name of the domain controller (for example, DC01.DOMAIN.COM) must appear in one of the following places:
  - The Common Name (CN) in the Subject field.
  - DNS entry in the Subject Alternative Name extension.
- The certificate was issued by a Certificate Authority (CA) that the domain controller and the LDAPS clients trust. Trust is established by configuring the clients and the server to trust the root CA to which the issuing CA chains.
- You must use the Schannel cryptographic service provider (CSP) to generate the key.

> **Note:** Regardless of what approach one chooses to generate the certificates, the above requirement has to be fulfilled for LDAP over SSL/TLS to be enabled in the Active Directory server.

Generating certificates

There are several ways for generating certificates for an Active Directory server you may choose depending on the particular security policy of your environment. What is crucial is to follow the requirements for a Microsoft Active Directory LDAP server.

Importing certificates to the Active Directory server

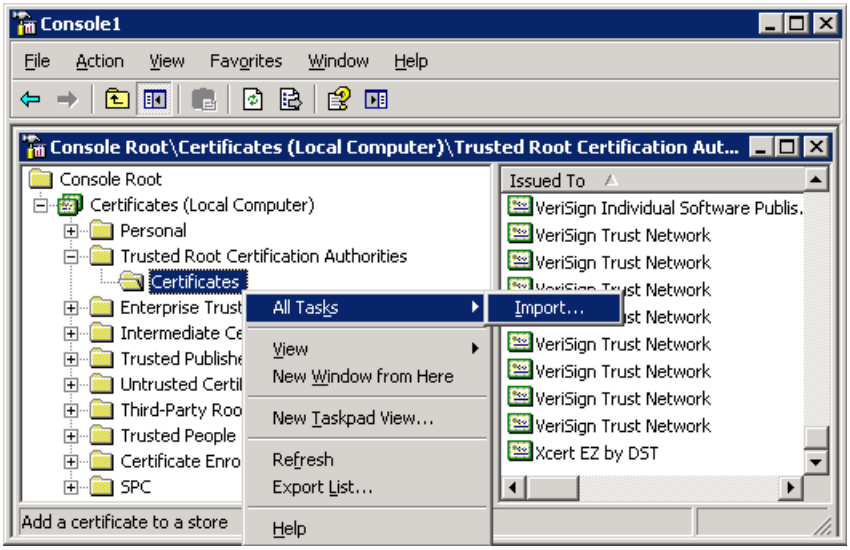
Once the certificates are generated, follow these steps to import them to the Active Directory server.

Make ready the CA (either commercial or self-signed) certificate and the server certificate so that it can be accessed from the Active Directory server. You can use any name you want during the certificate generation process. For example (ca.crt and server.crt).

1. Login to the Active Directory server as an Administrator.
2. To open the **Certificates** snap-in, click the Windows Start button, and type `MMC` into the search bar.
3. Click **Add...** to open the **Add Standalone** snap-in dialog.
4. In the **Add Standalone** Snap-In dialog, select **Certificates**, then press **Next**.
5. Select Computer account, then press **Next**.
6. Select Local computer, then press **Next**.
7. To close the **Add Standalone Snap-IN**, click **Close**.
8. When you have completed the certificate creation, click **Close**.

Import CA Certificate

1. Login to the Active Directory server as an Administrator.
2. To open the **Certificates** snap-in, click the Windows Start button, and type **MMC** into the search bar.
3. Expand the **Certificates** node under **Trusted Root Certification Authorities**.
4. In the **Welcome to the certificate Import Wizard** screen, right-click on the **Certificates** node, select **All Tasks** > **Import...**, and click Next.
5. In the **File to Import** screen, browse to the location where the CA certificate `ca.crt` is saved, and click **Next**.
6. In the **Certificate Store** screen, click **Next** and then **Finish**, see the example below.



7. Expand the Certificates node under Personal, and follow the import wizard to import the server certificate generated during the certificate generation process.
8. Restart the Active Directory server, for the changes to take effect.

Verifying the LDAPS connection

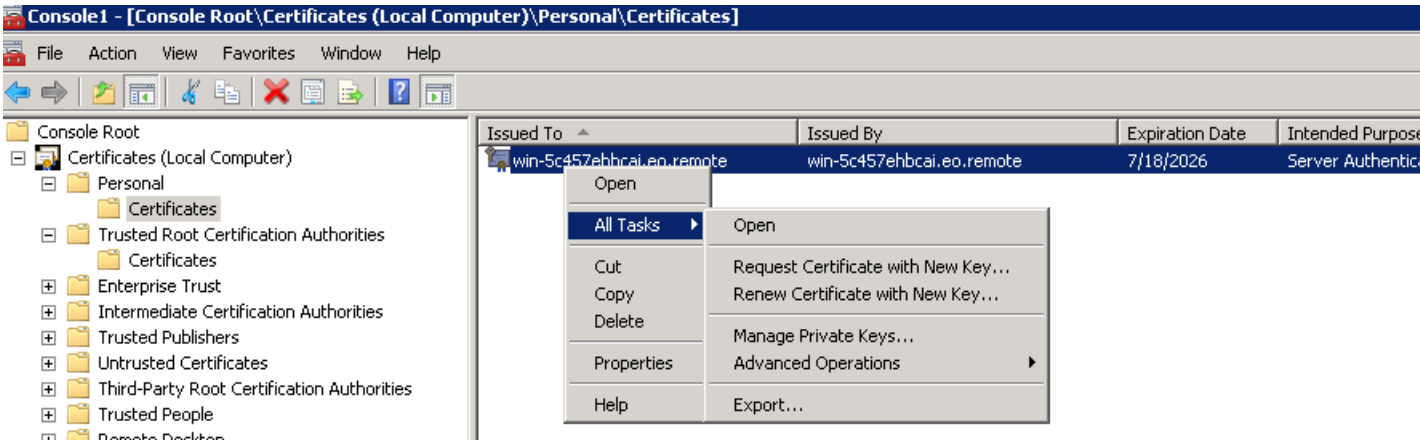
After a certificate is installed, follow these steps to verify that LDAPS is enabled:

1. Start the Active Directory Administration Tool (`Ldp.exe`).
2. On the **Connection** menu, click **Connect**.
3. Type the name of the domain controller to which you want to connect (This is the CN value in the 'Subject' section of the server certificate).
4. Type 636 as the port number.
5. Click **OK**.
   Dn:RootDSE information should print in the right pane, indicating a successful connection.

Exporting the LDAP server certificate for EngageOne Server use

This section describes how to export the Active Directory LDAP server PKCS12 (.PFX) certificate for EngageOne use. Assuming the **Certificates Snap-in** window above is still open, expand the **Certificates** node under **Personal**.

1. In the details pane, click the server certificate imported previously.
2. On the **Action** menu, point to **All Tasks**, and then click **Export**.

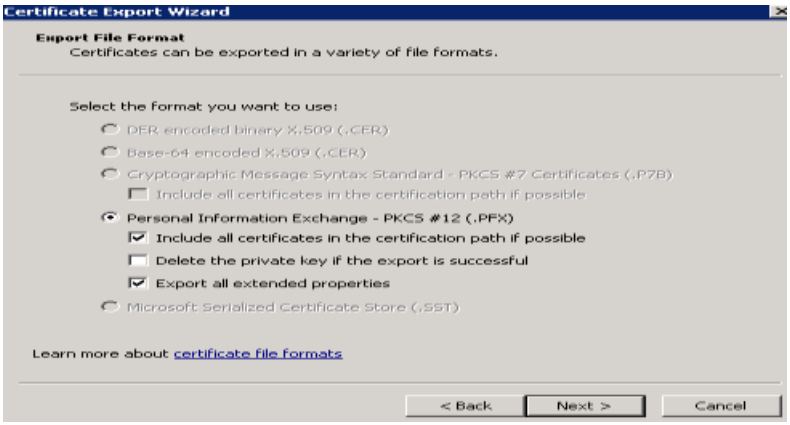


3. In the Certificate Export Wizard, click **Yes, export the private key**.

   Note: This option will appear only if the private key is marked as exportable and you have access to the private key.

4. In the next window, do the following:

   - To include all certificates in the certification path, select the **Include all certificates in the certification path if possible** check box.
   - To export the certificate's extended properties, select the **Export all extended properties** check box.



5. Click **Next**.
6. In **Password**, type a password to encrypt the private key you are exporting. In **Confirm password**, type the same password again, and then click **Next**.

   Note: This password will be used later in the “Configure Engage One Suite LDAP over SSL/TLS” section.

7. In **File name**, type a file name and path for the PKCS #12 file that will store the exported certificate and private key. Click **Next**, and then click **Finish**.

Configure EngageOne Server to use LDAP over SSL/TLS

This section describes how to import LDAP server’s PKSC#12(.PFX) certificate in to Engage One Server PKCS#12 store.

Engage One Server (EOS) uses a centralized “Personal Information Exchange” PKCS#12 (.PFX) trust/key store type. To enable LDAP over SSL/TLS the PKCS#12(.pfx) LDAP server certificate exported above needs to be merged into the EOS PKCS#12(.PFX) certificate store.

There are different tools to import one PKCS#12(.PFX) file in to another PKCS#12(.PFX) store, in the example below the Java keytool is used to demonstrate this step.

Make sure the LDAP server PKCS#12(.pfx) and the EOS PKCS#12(.PFX) files are accessible for the java keytool utility.

```
keytool -importkeystore -srckeystore <the-ldap-server-pksc12-cert-dot-pfx>
 -
srcstoretype pkcs12 -srcstorepass
<password-used-to-export-ldap-server-pkcs12-
certificate> -destkeystore
<EOS-centralized-PKCS12-dot-pfx-trust-key-store-filename>
-deststoretype pkcs12 -deststorepass <password-used-for-EOS-pfx-file>
```

Note: Use the `tls.version` property in your deploy.properties file to define the TLS version implemented in your environment; permissable values for this property are:

- TLSv1
- TLSv1.1
- TLSv1.2

Updating deploy.properties

In EOS once the LDAP server PKCS#12(.PFX) file successfully imported into the EOS PKCS#12(.pfx) trust/key store the following LDAP related properties need to be updated :

```
ldap.port=636
ldap.tls.enabled=true
ldap.url=ldaps://{LDAP-server-fully-qualified-name}:636
```

> **Note:** The `{LDAP-server-fully-qualified-name}` value should be the fully qualified domain name (FQDN) specified in the server certificates `Subject: CN` or `Subject Alternative Name:DNS Name`.

Attention: Make sure to consult the EOS Installation and Management documents “Certificate Configuration” guidelines to complete the enabling process.

Interactive Editor - OS and Network Infrastructure recommendations

The EngageOne Compose Interactive Editor uses WebSocket (IETF RFC 6455) communication protocol to provide full-duplex communication between browser and desktop applications. This communication is established via the WebSocket Relay Service which resides on the EngageOne Compose Core bundle. WebSockets are widely used and all browsers, http servers, reverse proxy servers and network devices support it. Depending on the deployment architecture of your EngageOne Compose suite and network architecture used it may be necessary to take certain actions to ensure your system operates in the best possible way. This section describes the areas that should be examined in case of communication/network issues that may be encountered when using the EngageOne Compose Interactive Editor.

Proxy servers

- If proxy servers are going to be present on the network infrastructure you may need to specifically configure them to handle WebSocket traffic.

  The following considerations should be taken into accounrt:

  - Long-lived connections (WebSockets) handling configuration. Proxy servers may choose to close streaming or idle WebSocket connections, because they appear to be trying to connect with an unresponsive HTTP server.
  - Proxy server caching configuration. Proxy servers may also buffer unencrypted HTTP responses and so introducing unpredictable latency during HTTP response streaming (WebSocket handshake).
  - Enabling `HTTP CONNECT` method on a Proxy server. This will be required for the WebSocket handshake to be successful as the browser client sends the `HTTP CONNECT` method when explicit Proxy server is being configured in the browser.
  - When a proxy server forwards a request to the (WebSocket) server, it is expected to strip off certain headers, including the `Connection:` header. Therefore, a well-behaved transparent proxy server will cause the WebSocket upgrade handshake to fail almost immediately.

- Hop-by-Hop Upgrade → During the WebSocket handshake, the `Connection: Upgrade` header is sent to the WebSocket server. If the proxy server has to participate in the upgrade mechanism, additional proxy server configuration would be required, because a hop-by-hop transport is used; the Upgrade sent from the proxy server to the browser is only good for that one hop, and the proxy server must send its own Upgrade header to handle the next hop from the proxy server to the WebSocket server (or to another intermediary server). In addition the proxy server must stop processing the request as HTTP.
- Use Web Socket's Secure with TLS encryption `(wss://)` to connect to Relay Service (EngageOne Compose Core bundle). Because the network traffic will be encrypted it should significantly increase the chances of a WebSocket connection to be established. This is due to the fact that there might be transparent proxy servers used in the network (those that are not explicitly declared in the browser). These servers will let the encrypted traffic through which should allow for successful WebSocket connection.
- Some HTTP proxy servers may restrict ports or allow access only to specific, authorized servers. A WebSocket server (EngageOne Compose Core bundle) that is used in such a scenario must be added to the white list of servers for connections to be successful.

Load Balancing

Types of Load Balancing

1. **TCP (Layer-4) load-balancing routers** should work well with WebSockets, because they have the same connection profile: connect once up front and stay connected, rather than the HTTP document transfer request-response profile.
2. **HTTP (Layer-7) load-balancing routers** expect HTTP traffic and can easily get confused by WebSocket upgrade traffic. For that reason, Layer 7 load balancing routers may need to be configured to be explicitly aware of WebSocket traffic.

Please consider which type of Load Balancing is being used in the network infrastructure where EngageOnne Compose suite is deployed and make appropriate actions to allow WebSocket traffic.

Load Balancer TCP Idle Timeout

The EngageOne Compose Interactive Editor maintains connection with the browser client by sending a heartbeat frame every 30 seconds. This mechanism provides reliable way of keeping open WebSocket connection between the editor and the browser client in case of an end-user inactivity. In order for this mechanism to work, the TCP Idle Timeout must greater than 30 seconds on Load Balancer (or Proxy server).

Operating system adjustments running EngageOne Compose Core bundle

Each WebSocket connection is assigned an ephemeral port (short-lived endpoint), by the operating system on which EngageOne Compose Core bundle is running. The operating system selects the port number from a predefined range (1024 - 655354) and releases the port after the related TCP connection terminates. On Windows Server, by default, the system can create approximately 16,000 ephemeral ports that run concurrently. Each Linux distribution has different default values set depending on the distribution but also on type of the user being used to run EngageOne Compose services and the system-wide resource limits assigned to it.

The amount of system-wide resources including number of ephemeral ports must be adjusted accordingly as instructed by each operating system provider depending on:

- The expected number of concurrent users predicted to work on the system and actively creating communications.
- The amount of EngageOne Compose Core bundle installations.

Account lockout policy configuration

You can, if required, configure the account lockout policy by setting the appropriate option in your system's deploy.properties file.

security.account.lockout.options - is the deploy.properties option that governs the lockout policy and is used to set the policy during installation of the security bundle.

This option allows you to configure six values separated by semicolons, as indicated below:

```
security.account.lockout.options=value_1;value_2;value_3;value_4;value_5;value_6
```

where each value is defined, as follows:

Value	Description
value_1	Lockout policy state: • *true* - sets Lockout Policy State to enabled • *false* - sets Lockout Policy State to disabled
value_2	Login Failure Count - this integer value determines the number of failed logins before locking out the account.
value_3	Lockout User Warning - this integer value determines the number of failed logins before a warning is issued.
value_4	Failure Storing Time. Time (in minutes) during which failed login attempts are stored in memory. Accepted only integer values.
value_5	Lockout Duration. Determines time (in minutes) of the account lockout. Accepted only integer values.

Value	Description
value_6	Lockout Duration Multiplier. Specifies the value by which the Lockout Duration Time will be multiplied in case of violating the policy more than one time. For example if policy defines Lockout Duration to 3 (minutes), Lockout Duration Multiplier to 2 and it is violated 2 times in the row (in specified in Failure Storing Time option time) - first violation lock would last 3 minutes, second one - 6 minutes (Lockout Duration * Multiplier). Accepted only integer values. The account lockout configuration is stored in the security bundle memory to ensure there is no impact on your LDAP service. security.account.lockout.options is an optional property, which is set by default as follows: false;10;5;3;3;2 which means:

Example

```
security.account.lockout.options=true;10;5;10;5;2
```

In this example, the following lockout settings are enforce:

- account lockout is enforced
- 10 failed login attempts are allowed before lockout is enforced
- a warning is issued after 5 failed login attempts
- the number of failures is stored for 10 minutes before being reset
- the duration of the lockout is 5 minutes
- the lockout duration is doubled on each subsequent lockout

Note: the default setting is as follows:

```
security.account.lockout.options=false;10;5;3;3;2
```

Refer to the Account unlocking section in the EngageOne Server Administration Guide for details on how to unlock a locked account.

HTTP Strict Transport Security (HSTS)

The HTTP Strict Transport Security (HSTS) mechanism guarantees protection for MitM attacks such as protocol downgrading, or cookie hijacking. By default, HSTS is enabled, but in certain configurations, this security may cause connectivity issues, for example, when a user or other mechanism tries to connect to the bundle over HTTP protocol, but connections between these two instances were previously made over HTTPS. For this reason, an option to disable the HSTS mechanism is provided.

To disable HSTS, edit the properties file for your selected bundle(s), located as follows:

```
<bundle>/conf/bundle.properties
```

The property responsible for HSTS is:

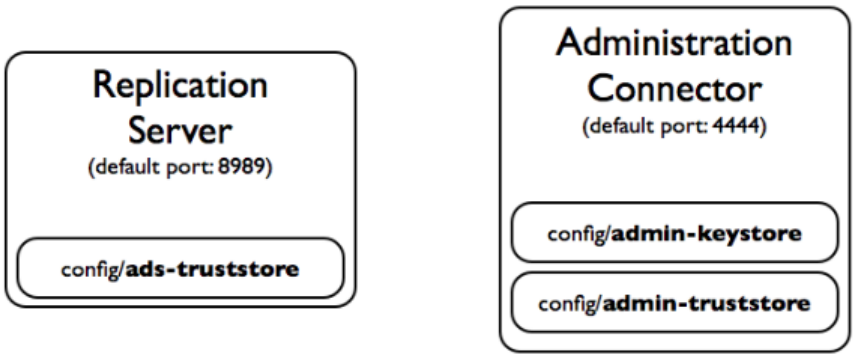
```
http.strict.transport.security.header.enabled
```

To disable HSTS change the *true* default setting to *false*.

OpenDS Certificate Configuration

Introduction

OpenDJ uses keystores (for private keys) and truststores (for public, signed certificates). The EngageOne implementation of OpenDJ uses two sets of keystores as shown in the diagram:



Points to note:

- By default the keystores are located in:

```
{security_bundle_dir}/conf/OpenAM/opends/config
```

- The keystore and truststore hold keys for securing connections with client applications.
- The admin-keystore and admin-truststore hold keys for securing administrative connections, such as those used when connecting with the dsconfig command.
- The ads-truststore holds keys for securing replication connections with other OpenDJ servers in the replication topology.

Each keystore has a specific purpose, as follows:

admin-truststore

This Java Keystore holds the private key and administrative certificate for the server, admin-cert. This key pair is used to protect communications on the administration port. The password, stored in admin-keystore.pin, is also the key password for admin-cert.

admin-truststore

This Java Keystore holds a copy of the administrative certificate, admin-cert. The password is the same as for the admin-keystore, the string in the admin-keystore.pin.

ads-truststore

This Java Keystore holds public key certificates of all servers replicating with the current server. It also includes the ads-certificate key pair of the current server. The password is stored in ads-truststore.pin

The sections that follow describe the steps to change the default self-signed certificates with CA signed certificates for both admin-keystore and ads-keystore.

Admin keystore configuration

The recommended approach to change the certificate used by admin-keystore is to replace the default certificate with CA signed certificate

Replacing with CA SignedCertificate on OpenAM Node

1. Login to OpenAM node
2. Navigate to:

```
/opt/engageone/server/security/conf/OpenAM/opends/config
```

3. Take a backup of the default admin-keystore , admin-truststore files.
4. Run the following command to delete the default certificate from admin-keystore and admin-truststore.

```
keytool -delete -alias admin-cert -keystore admin-keystore
keytool -delete -alias admin-cert -keystore admin-truststore
```

When prompted enter the password from admin-keystore.pin file.

5. Verify the certificate is deleted from the keystore , run the command:

```
keytool –list –v –keystore admin-keystore
```

When prompted enter the password from admin-keystore.pin.

```
[ec2-user@cvslab-eonode1 config]$ sudo keytool -delete -alias admin-cert -keysto
re admin-keystore
Enter keystore password:
[ec2-user@cvslab-eonode1 config]$ keytool -list -v -keystore admin-keystore
Enter keystore password:
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 0 entries
```

6. Obtain signed key pair in PKCS12 format.
7. Import the keypair into the keystore:

```
keytool -importkeystore -srckeystore {src_keypair_file} -destkeystore

admin-keystore - srcstoretype pkcs12 -alias admin-cert
```

8. Import the root certificate to the truststore:

```
keytool -importcert -alias admin-cert -keystore admin-truststore-file
 root.cer
```

9. Restart the security bundle
10. Repeat steps 1 through 9 for all nodes with security bundle. If the same certificate is used for all nodes you can skip steps 3 through 8 and copy admin-keystore, admin-truststore and admin-keystore.pin from the first node.

Replication keystore configuration

Disable replication on OpenAM Nodes

EngageOne Compose enables replication as part of the configuration. To configure custom certificates for replication, stop the replication between the OpenAM nodes

1. Login to OpenAM Node 1 and navigate to:

```
{security_bundle_dir}/conf/OpenAM/opends/bin
```

2. Run the command

```
./dsreplication
```

3. Select option 2 to disable replication.

```
[ec2-user@cvslab-eonode1 bin]$ sudo ./dsreplication
What do you want to do?

    1)  Enable Replication
    2)  Disable Replication
    3)  Initialize Replication on one Server
    4)  Initialize All Servers
    5)  Pre External Initialization
    6)  Post External Initialization
    7)  Display Replication Status
    8)  Purge Historical
    9)  Re-synchronizes the change-log changenumber on one server with the
        change-log changenumber of another.

    c)  cancel

Enter choice: 2
```

4. Accept the default setting for hostname,

```
Directory server hostname or IP address [cvslab-eonode1]:
```

5. Accept the default for server administration port,

```
Directory server administration port number [4444]:
```

6. Enter the Global Administrator UserID as `cn=Directory Manager`

```
Global Administrator User ID, or bind DN if no Global Administrator is defined
[admin]: cn=Directory Manager
```

7. Enter the password for the users : `<amadmin password>`
8. Accept the default value `Yes`,

```
You must choose at least one base DN to be disabled.
Disable replication on base DN dc=openam,dc=forgerock,dc=org? (yes / no) [yes]:
You have chosen to disable all the replicated base DNs in the server
'cvslab-eonode1:4444'.  Do you want to disable also the replication port
'50889'? (yes / no) [yes]:
```

9. Accept the default value `Yes`,

```
Disabling replication will make the data under the selected base DNs not to be
synchronized with other servers any more.  Do you want to continue? (yes / no)
[yes]:
```

10. Replication is disabled,
11. Repeat the steps on all nodes with security bundle installed.

Viewing and deleting the default certificates on OpenAM Nodes

1. Login to OpenAM
2. Navigate to:

```
{security_bundle_dir}/conf/OpenAM/opends/config
```

3. Run the command:

```
keytool -list -v -keystore ads-truststore
```

When prompted for password, enter the password from the file ads-truststore.pin,

```
[ec2-user@cvslab-eonode2 config]$ sudo keytool -list -v -keystore ads-truststore
Enter keystore password:
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 3 entries

Alias name: 9d92c827661d576a660ace47c368611a
Creation date: Mar 6, 2018
Entry type: trustedCertEntry

Owner: CN=cvslab-eonode2.cvstestlab.corp.local, O=OpenDJ RSA Certificate
Issuer: CN=cvslab-eonode2.cvstestlab.corp.local, O=OpenDJ RSA Certificate
Serial number: 3e49d556
Valid from: Tue Mar 06 20:19:45 UTC 2018 until: Mon Mar 01 20:19:45 UTC 2038
Certificate fingerprints:
         MD5:  9D:92:C8:27:66:1D:57:6A:66:0A:CE:47:C3:68:61:1A
         SHA1: 1F:8E:1F:8B:90:E2:0A:84:5B:EF:41:7A:B2:2C:51:2C:FF:56:EC:0D
         SHA256: F7:47:BD:54:CA:7B:56:07:D6:2C:62:20:C1:17:55:0B:C7:B9:5C:12:0B:
2B:09:65:8F:86:9E:B6:2A:A9:88:C6
Signature algorithm name: SHA1withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
```

4. You will find a number of entries, 1 private key and 1 trusted entry for each security node
5. Copy the Alias name for the current node trusted entry, it is lowercase MD5 fingerprint of private key without colons.

```
Alias name: 1c1c998cfe214546dee8945ff8b583f5
Creation date: Mar 6, 2018
Entry type: trustedCertEntry

Owner: CN=cvslab-eonode1.cvstestlab.corp.local, O=OpenDJ RSA Certificate
Issuer: CN=cvslab-eonode1.cvstestlab.corp.local, O=OpenDJ RSA Certificate
Serial number: 4c4f0b16
Valid from: Tue Mar 06 20:14:59 UTC 2018 until: Mon Mar 01 20:14:59 UTC 2038
Certificate fingerprints:
         MD5:  1C:1C:99:8C:FE:21:45:46:DE:E8:94:5F:F8:B5:83:F5
         SHA1: 84:FD:D9:1A:97:3A:78:72:49:95:92:BF:71:86:D2:87:1F:14:61:0C
         SHA256: 86:3A:DA:B1:4F:45:92:F3:E6:B5:98:64:78:58:A9:30:0D:07:67:90:75:
20:C1:4A:F3:2E:AF:A5:51:71:BD:A8
Signature algorithm name: SHA1withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
```

6. Open the file admin-backend.ldif and verify if the alias name matches the ds-cfg-key-id,

```
dn: ds-cfg-key-id=1C1C998CFE214546DEE8945FF8B583F5,cn=instance keys,cn=admin data
objectClass: top
objectClass: ds-cfg-instance-key
ds-cfg-key-id: 1C1C998CFE214546DEE8945FF8B583F5
ds-cfg-public-key-certificate;binary:: MIIDHDCCAgSgAwIBAgIETE8LFjANBgkqhkiG9w0BAQUFADBQM
```

7. Navigate to: `$OpenDS_HOME/bin`

and run the command:

```
./ldapdelete --port 50389 --bindDN "cn=Directory Manager" --bindPassword
{amadmin_password} "ds-cfg-key-id=
1C1C998CFE214546DEE8945FF8B583F5,
cn=instance keys,cn=admin data"
```

```
[ec2-user@cvslab-eonode1 bin]$ sudo ./ldapdelete --port 50389 --bindDN "cn=Directory Manager" --bindPassword CVSL
E8945FF8B583F5,cn=instance keys,cn=admin data"
Processing DELETE request for ds-cfg-key-id= 1C1C998CFE214546DEE8945FF8B583F5,cn=instance keys,cn=admin data
DELETE operation successful for DN ds-cfg-key-id= 1C1C998CFE214546DEE8945FF8B583F5,cn=instance keys,cn=admin data
[ec2-user@cvslab-eonode1 bin]$
```

8. Navigate to: `{security_bundle_dir}/ conf/OpenAM/opends/ configbackup ads-truststore` and remove all entries from it, or create new one with the same name and password.
9. Repeat steps 1 through 8 on all nodes with the security bundle installed.

Creating new keystore and certificates

1. Navigate to: `{security_bundle_dir}/ conf/OpenAM/opends /config`
2. Get the certificate signed by CA in PKCS12 format.
3. Run the command to import the signed private key to the keystore:

```
keytool -importkeystore -srckeystore {src_keypair_file} -destkeystore

ads-truststore -srcstoretype pkcs12 -alias ads-certificate
```

When prompted enter the password from the file ads-truststore.pin file.

4. View the private key pair in the keystore to copy the MD5 fingerprint value:

```
[ec2-user@cvslab-eonode1 config]$ sudo keytool -list -v -keystore
Enter keystore password:
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

Alias name: ads-certificate
Creation date: Mar 16, 2018
Entry type: PrivateKeyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=CVS Test Lab
Issuer: CN=cvstestlab-CVSLAB-SQL01-CA, DC=cvstestlab, DC=corp, DC=
Serial number: 73000000061426d74e9fa70acc000000000006
Valid from: Fri Mar 16 15:10:17 UTC 2018 until: Sun Mar 15 15:10:1
Certificate fingerprints:
         MD5:  25:D7:36:6A:05:B7:52:29:13:F4:4A:0B:F1:D1:36:AD
         SHA1: 11:63:23:EC:9C:C0:05:BD:5C:E5:0C:05:E3:2C:18:CF:FA:
         SHA256: 19:DE:03:8B:88:AB:C9:53:C4:6C:A1:4D:6F:2D:E2:13:F
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
```

5. Define a new alias name using the MD5 fingerprint and remove the ':' and change the alphabetic characters to lower case

   The alias is `25d7366a05b7522913f44a0bf1d136ad`

6. Save the alias and root certificate for later use.
7. Repeat steps 1 through 6 for each node with the security bundle installed.

Adding trusts between nodes

The replication works on trusting the certificate of Node1 on Node 2, and vice-versa. This requires importing the certificates with the appropriate alias names on each node. In the previous procedure we have a set of aliases and corresponding certificates, one for each node. We will import all certificates as a trusted entry in ads-truststore on each node.

1. Login to OpenAM node
2. Navigate to: `{security_bundle_dir}/conf/OpenAM/opends/config`
3. Run the commands to import all certificates gathered in point 6 of previous section with proper aliases

```
keytool -importcert -alias {mdf_fingerprint_alias} -keystore ads
- truststore -file {root_certificate_file}
```

4. Repeat on all nodes with an installed security bundle.

Enable replication

1. Restart the `eos.security` service on node 1 and node 2
2. Login to OpenAM node
3. Navigate to: `$Opends_HOME/bin`
4. Run the command: `./dsreplication`
5. Select option 1,

```
[ec2-user@cvslab-eonode1 bin]$ sudo ./dsreplication
What do you want to do?

    1)  Enable Replication
    2)  Disable Replication
    3)  Initialize Replication on one Server
    4)  Initialize All Servers
    5)  Pre External Initialization
    6)  Post External Initialization
    7)  Display Replication Status
    8)  Purge Historical
    9)  Re-synchronizes the change-log changenumber on one server with the
        change-log changenumber of another.

    c)  cancel

Enter choice: 1
```

6. Accept default for Directory Server,

```
Directory server hostname or IP address [cvslab-eonode1]:
```

7. Accept default value,

```
Directory server administration port number [4444]:
```

8. Enter `cn=Directory Manager`,

```
Global Administrator User ID, or bind DN if no Global Administrator is defined
[admin]: cn=Directory Manager
```

9. Enter the amadmin password,

```
Password for user 'cn=Directory Manager':
```

10. Accept default,

```
Replication port for the first server (the port must be free) [8989]:
```

11. Type `yes` for encryption,

```
Do you want replication to use encrypted communication when connecting to
replication port 8989 on the first server? (yes / no) [no]: yes
```

12. Enter the OpenAM node to hostname or IP,

```
>>>> Specify server administration connection parameters for the second server

Directory server hostname or IP address [cvslab-eonode1]: cvslab-eonode2
```

13. Accept the default,

```
Directory server administration port number [4444]:
```

14. Select option 3 or 1,

```
How do you want to trust the server certificate?

    1)  Automatically trust
    2)  Use a truststore
    3)  Manually validate

Enter choice [3]:
```

15. Enter `cn=Directory Manger`,

```
Global Administrator User ID, or bind DN if no Global Administrator is defined
[admin]: cn=Directory Manager
```

16. Enter the amadmin password,

```
Password for user 'cn=Directory Manager':
```

17. Accept the default,

```
Replication port for the second server (the port must be free) [8989]:
```

18. Type `yes` for encrypted traffic,

```
Do you want replication to use encrypted communication when connecting to
replication port 8989 on the second server? (yes / no) [no]: yes
```

19. Accept the default,

```
You must choose at least one Base DN to be replicated.
Replicate base DN dc=openam,dc=forgerock,dc=org? (yes / no) [yes]:
```

Replication is enabled,

```
Establishing connections ..... Done.
Checking registration information ..... Done.
Configuring Replication port on server cvslab-eonode1:4444 ..... Done.
Configuring Replication port on server cvslab-eonode2:4444 ..... Done.
Updating replication configuration for baseDN dc=openam,dc=forgerock,dc=org on
server cvslab-eonode1:4444 .....Done.
Updating replication configuration for baseDN dc=openam,dc=forgerock,dc=org on
server cvslab-eonode2:4444 .....Done.
Updating registration configuration on server cvslab-eonode1:4444 ..... Done.
Updating registration configuration on server cvslab-eonode2:4444 ..... Done.
Updating replication configuration for baseDN cn=schema on server
cvslab-eonode1:4444 .....Done.
Updating replication configuration for baseDN cn=schema on server
cvslab-eonode2:4444 .....Done.
Initializing registration information on server cvslab-eonode2:4444 with the
contents of server cvslab-eonode1:4444 .....Done.
Initializing schema on server cvslab-eonode2:4444 with the contents of server
cvslab-eonode1:4444 .....Done.

Replication has been successfully enabled.  Note that for replication to work
you must initialize the contents of the base DNs that are being replicated
(use dsreplication initialize to do so).
```

20. Repeat the steps above on all nodes with the security bundle installed. Ensure you enter the primary security node hostname or IP address used in step12

Initialize the DN

1. Open OpenAM node 1 run the command: `./dsreplication initialize`
2. Accept the default value,

```
Directory server hostname or IP address [cvslab-eonode1]:
```

3. Accept the default value,

```
Directory server administration port number [4444]:
```

4. Accept the default value admin,

```
Global Administrator User ID [admin]:
```

5. Enter the amadmin password,

```
Password for user 'admin':
```

6. Enter the hostname of OpenAM replica node

```
>>>> Specify server administration connection parameters for the destination
server

Directory server hostname or IP address [cvslab-eonode1]: cvslab-eonode2
```

7. Accept the default value,

```
Directory server administration port number [4444]:
```

8. Select option 1 or 3,

```
How do you want to trust the server certificate?

    1)  Automatically trust
    2)  Use a truststore
    3)  Manually validate
```

9. Accept the default value,

```
You must choose at least one base DN to be initialized.
Initialize base DN dc=openam,dc=forgerock,dc=org? (yes / no) [yes]:
```

10. Accept the default,

```
Initializing the contents of a base DN removes all the existing contents of
that base DN.  Do you want to remove the contents of the selected base DNs on
server cvslab-eonode2:4444 and replace them with the contents of server
cvslab-eonode1:4444? (yes / no) [yes]:
```

    Initializing is complete

```
Initializing base DN dc=openam,dc=forgerock,dc=org with the contents from
cvslab-eonode1:4444:
260 entries processed (42 % complete).
608 entries processed (100 % complete).
Base DN initialized successfully.
```

11. Repeat steps 1-10 providing the address in point 6 for each node with the security bundle installed.

Replication Status

1. On any OpenAM node, navigate to: `{security_bundle_dir}/conf/OpenAM/opends /bin`
2. Run the command: `./dsreplication`
3. Select Option 7,

```
    1)  Enable Replication
    2)  Disable Replication
    3)  Initialize Replication on one Server
    4)  Initialize All Servers
    5)  Pre External Initialization
    6)  Post External Initialization
    7)  Display Replication Status
    8)  Purge Historical
    9)  Re-synchronizes the change-log changenu
        change-log changenumber of another.

    c)  cancel

Enter choice: 7
```

4. Accept default,

```
Directory server hostname or IP address [cvslab-eonode1]:
```

5. Accept default,

```
Directory server administration port number [4444]:
```

6. Accept default admin,

```
Global Administrator User ID [admin]:
```

7. Enter the amadmin password,

```
Password for user 'admin':
```

8. Enter option 2,

```
Do you trust this server certificate?

    1)  No
    2)  Yes, for this session only
    3)  Yes, also add it to a truststore
    4)  View certificate details
```

9. You should see the replication status,

```
Suffix DN                : Server              : Entries : Replication enabled : DS ID : RS ID : RS Port (1) : M.C. (2) : A.O.M.C. (3) : Security (4)
-------------------------:---------------------:---------:---------------------:-------:-------:-------------:----------:--------------:-------------
dc=openam,dc=forgerock,dc=org : cvslab-eonode1:4444 : 609     : true                : 14357 : 23339 : 8989        : 0        :              : true
dc=openam,dc=forgerock,dc=org : cvslab-eonode2:4444 : 609     : true                : 26091 : 19941 : 8989        : 0        :              : true

[1] The port used to communicate between the servers whose contents are being
replicated.
[2] The number of changes that are still missing on this server (and that have
been applied to at least one of the other servers).
[3] Age of oldest missing change: the date on which the oldest change that has
not arrived on this server was generated.
[4] Whether the replication communication through the replication port is
encrypted or not.
```

10. Security `true` indicates that all replication traffic is encrypted using TLS certificate.

Validation

After the services restart

Check for successful login to:

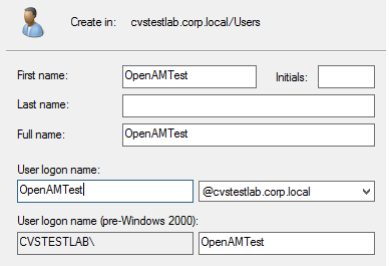
- OpenAM
- EngageOne Interactive
- EngageOne SOAP services

Check for successful OpenAM replication, to test:

- Make a change in the directory (add user, delete user),
- Verify the change is available on both nodes by using the node URL directly.

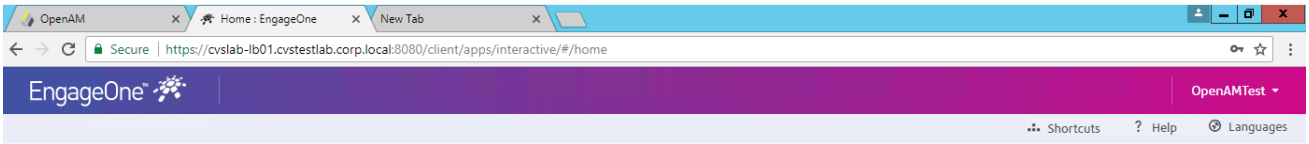
Example:

1. Add a new user in Active Directory,



2. Save the changes,
3. Login to the first OpenAM node , click on EngageOne realm->Subjects,
4. Verify the new user is available,

5. Login to second OpenAM node URL, and verify the user is available,
6. Login to EngageOne with the new user credentials.



References

https://backstage.forgerock.com/knowledge/kb/article/a33131480

https://backstage.forgerock.com/docs/opendj/3.5/admin-guide/#chap-change-certs

Log collector

Introduction

The Log Collector is a tool which collects log files from every bundle in a particular EO Compose node, once collected from all nodes, Log Collector zips all logs into a single file and copies the zip file to the `active-drive\logCollector`. The tool is located in `<installation location>\bin` folder of every bundle and it collects files from the following folders:

The security bundle:

- `conf\OpenAM\OpenAM\debug`
- `conf\OpenAM\OpenAM\log`
- `conf\OpenAM\opends\logs`
- `logs`

Remaining bundles:

- `logs`

Note that the `deploy.properties` file is also copied without passwords.

Running the log-collector script

Function	The Log collector script executes log collection for the current node. The log-collector behavior can be customized by supplying the appropriate parameters, as follows:
Command Line Interface	
Script	`prompt> log-collector {parameters}`

Example	`prompt> log-collector -p deploy.properties`
	
	`prompt> log-collector -p deploy.properties -d 3 -n node1`

Prameters	
`-p, -properties <deployPropertiesPath>`	*REQUIRED* Path directly to `deploy.properties` file used during bundle installation.
`-d, -numDaysRetention <daysNumber>`	Include in the copying process, all log files created later than the specified number of days from the current date. For example, providing `-d 1` will gather today's and yesterday's logs.
`-f, -finalize`	Packages logs from `logCollector/work` directory on active-drive to one zip (or from `<outputPath>/work`, if the `-outputPath` parameter is provided)
`-h, -help`	Prints syntax and usage information for this log collection job.

`-n, -name <name>`	This parameter is used as a zip file name of all logs from a particular node. If it is not provided, the computer name is used.
`-o,-outputPath <outputFolderPath>`	Path to a shared folder where logs will be copied. All nodes must have access to this location. Default folder is on active-drive.
`-q, -quiet`	Do not display the logs on the console.

Additonal notes

Only the -p (properties) parameter is mandatory.
If the -o (outputPath) parameter is not provided, log files are copied to the `active-drive\logCollector\work\<bundle>` folder.
If the -o parameter is used to collect logs from node1, the same -o parameter must be used to gather logs from the remaining nodes.
If the -d (numDaysRetention) parameter is not provided, it defaults to -1, which means all existing logs are copied.
If the -n (name) parameter is not provided, the computer name/hostname is used as a zip file name.
If the -n parameter is to be used, it must be unique for each node.
The -f (finalize) parameter must be provided when executing the log-collector on the last node

Exmples

Recommended for standalone environments:

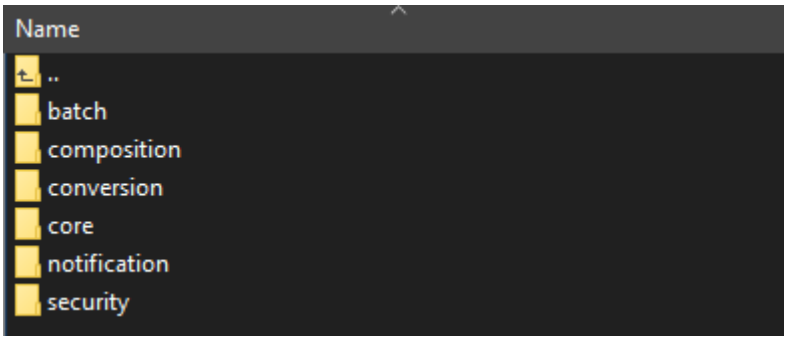
```
log-collector -p deploy.properties -o C:\logs -finalize
```

Recommended for clustered environments:

```
log-collector -p deploy.properties -n node1
log-collector -p deploy.properties -n node2
log-collector -p deploy.properties -n node3
log-collector -p deploy.properties -n node4 -finalize
```

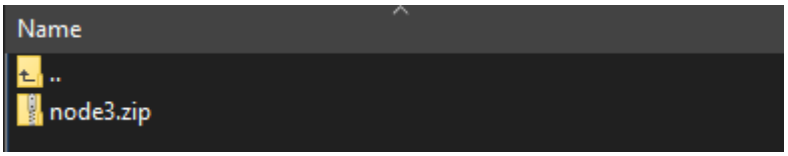
The Work folder structure

By default, logs of a single bundle from particular node are zipped as `<computer name>.zip` and copied to the `active-drive\logCollector\work\<bundle>` folder. After executing log-collector for node1, node2 and node3 (providing -n parameter for each), the following content can be found in `active-drive\logCollector\work` folder:

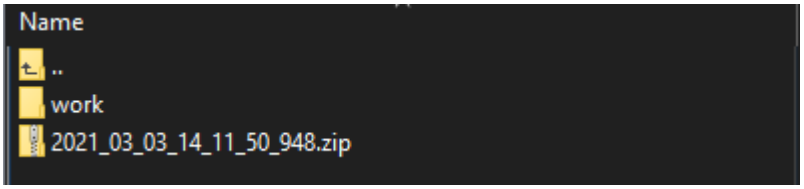


Content of subfolders of bundles installed on node1 and node2:

Content of subfolders of bundles installed on node3:



After executing log-collector for node4 and providing -finalize parameter, logs from node4 are copied and zipped in the same way as for previous nodes, the content of “work” folder is zipped into <system_date>_<system_time>.zip and placed in active-drive\logCollector, “work” folder content is then deleted:



The final zip file can be provided to your support team for further investigation.

Notices

Matra 0.8.2b (**http://matra.sourceforge.net/**) The contents of this documentation are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this documentation except in compliance with the License. You may obtain a copy of the License at **http://www.mozilla.org/MPL/**. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. Otherwise all product names are trademarks or registered trademarks of their respective holders.

This product contains Sycamore, version number 0.5.0, which is licensed under the MIT license. The license can be downloaded from **https://github.com/ryexley/sycamore/blob/master/dist/requester.js**. The source code for this software is available from **https://github.com/ryexley/sycamore**.

This product contains Underscore, version number 1.13.1, which is licensed under the MIT license. The license can be downloaded from **https://github.com/jashkenas/underscore/blob/master/LICENSE**. The source code for this software is available from **http://underscorejs.org/**.

This product contains Flowable, version number 6.4.2, which is licensed under the Apache license. The license can be downloaded from **https://github.com/flowable/flowable-engine/blob/master/LICENSE**. The source code for this software is available from **https://github.com/flowable/flowable-engine**.

This product contains Bootstrap, version number 3.4.1, which is licensed under the MIT license. The license can be downloaded from **http://getbootstrap.com/getting-started/#license-faqs**. The source code for this software is available from **http://getbootstrap.com/getting-started/#download.**.

This product contains Commons-Configuration, version number 1.10, which is licensed under the Apache license. The license can be downloaded from **http://www.apache.org/licenses/**. The source code for this software is available from **http://commons.apache.org/proper/commons-configuration/**.

This product contains jQuery, version number 3.5.1, which is licensed under the MIT license. The license can be downloaded from **https://jquery.org/license/**. The source code for this software is available from **http://jquery.com/download/**.

This product contains Knockout-AMD-Helpers, version number 0.7.4, which is licensed under the MIT license. The license can be downloaded from **https://github.com/rniemeyer/knockout-amd-helpers/blob/master/LICENSE**. The source code for this software is available from **https://github.com/rniemeyer/knockout-amd-helpers**.

This product contains Knockout, version number 3.5.1, which is licensed under the MIT license. The license can be downloaded from **https://github.com/knockout/knockout/blob/master/LICENSE**. The source code for this software is available from **http://knockoutjs.com/downloads/**.

This product contains bootstrap-datetimepicker, version number 4.17.49, which is licensed under the MIT license. The license can be downloaded from **https://github.com/myactionreplay/bootstrap-datetimepicker/blob/master/LICENSE**. The source code for this software is available from **https://github.com/myactionreplay/bootstrap-datetimepicker**.

This product contains Knockout-DelegatedEvents, version number 0.6.1, which is licensed under the MIT license. The license can be downloaded from **https://github.com/rniemeyer/knockout-delegatedEvents#license**. The source code for this software is available from **https://github.com/rniemeyer/knockout-delegatedEvents**.

This product contains Moment.js, version 2.29.1, which is licensed under the MIT license. The license can be downloaded from **https://github.com/moment/moment/blob/develop/LICENSE**. The source code for this software is available from **http://momentjs.com/**.

This product contains Quartz-Scheduler, version number 2.3.2, which is licensed under the Apache license. The license can be downloaded from **http://quartz-scheduler.org/**. The source code for this software is available from **http://quartz-scheduler.org**.

This product contains RequireJS Text, version number 2.0.15, which is licensed under the BSD and MIT licenses. The license can be downloaded from **https://github.com/requirejs/text/blob/master/LICENSE**. The source code for this software is available from **https://github.com/requirejs/text**.

This product contains RequireJS, version number 2.3.6, which is licensed under the BSD and MIT licenses. The license can be downloaded from **https://github.com/jrburke/requirejs/blob/master/LICENSE**. The source code for this software is available from **http://requirejs.org/docs/download.html**.

This product contains Apache ActiveMQ, version number 5.15.9 , which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/**.The source code for this software is available from **http://activemq.apache.org**

This product contains Apache NMS version 1.7.2, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/**.The source code for this software is available from: **http://activemq.apache.org/nms**

This product contains Apache Commons DBCP2, version number 2.6.0, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/**.The source code for this software is available from **https://commons.apache.org/proper/commons-dbcp/**.

This product contains OWASP Encoder, version number 1.2.2, which is licensed under the BSD license. The license can be downloaded from **https://opensource.org/licenses/BSD-3-Clause**. The source code for this software is available from **https://www.owasp.org/index.php/OWASP_Java_Encoder_Project**.

This product contains Narayan, version number 5.2.13.Final, which is licensed under the LGPL license, version number 2.1. The license can be downloaded from **http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt**. The source code for this software is available from **http://narayana.io/**.

This product contains Logback, version number 1.2.3, which is licensed under the EPL and LGPL licenses, version numbers 1.0 and 2.1. The license can be downloaded from **http://logback.qos.ch/license.html**. The source code for this software is available from **http://logback.qos.ch/**.

This product contains JBoss Weld, version number 3.1.0.Final, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/LICENSE-2.0**.The source code for this software is available from **http://weld.cdi-spec.org**.

This product contains Hibernate, version number 5.4.25.Final, which is licensed under the Apache and LGPL license, version numbers 2.0 and 2.1. The license can be downloaded from **http://hibernate.org/community/license/**. The source code for this software is available from **http://hibernate.org/orm/**.

This product contains Apache Tomcat, version number 9.0.43, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/LICENSE-2.0**. The source code for this software is available from **http://tomcat.apache.org/**.

This product contains Apache Procrun, version number 1.1.0, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/**.The source code for this software is available from **http://commons.apache.org/proper/commons-daemon/procrun.html**.

This product contains FasterXML Jackson, version number 2.9.8, which is licensed under the Apache license, version number 2.0. The license can be downloaded from **http://www.apache.org/licenses/**. The source code for this software is available from **https://github.com/FasterXML/jackson**.

This product contains Log4net. The license for log4net can be downloaded from **https://www.apache.org/licenses/LICENSE-2.0**. The source code for this software is available from **https://logging.apache.org/log4net/download_log4net.cgi**.

Support

Click **here** for full EngageOne Compose documentation and access to your peers and subject matter experts on the Knowledge community.

precisely

1700 District Ave Ste 300
Burlington MA 01803-5231
USA

www.precisely.com