

Escenic Content Engine
Installation Guide
5.6.13.183224

Table of Contents

1 Introduction	4
1.1 Installation Components	4
1.1.1 Escenic Components	4
1.1.2 Third-Party Components	5
1.2 Installation Types	5
1.3 Conventions Used in This Manual	6
1.3.1 Identifying Host Machines	6
1.3.2 Shell Commands	7
1.3.3 Standard File Hierarchy	7
2 Product Requirements	9
2.1 Required Supporting Software	9
2.2 Other Requirements	9
2.2.1 Shared Filesystem	9
2.2.2 SSH/FTP Server	10
2.2.3 Apache Ant	10
3 Installation Procedure	11
3.1 Install Java SE Development Kit (JDK)	11
3.2 Install Ant	11
3.3 Install Various Utilities	12
3.4 Create Escenic Users	12
3.5 Create Shared File System	13
3.6 Download Content Engine	14
3.7 Install Database	15
3.8 Install Application Server	16
3.9 Create Configuration Folder	19
3.10 Unpack Content Engine Components	19
3.11 Logging	20
3.12 Copy Solr Configuration	20
3.13 Initialize the Assembly Tool	21
3.14 Initialize the Bootstrap Layer	21
3.15 Create the Common Configuration Layer	21
3.16 Create Host Configuration Layers	23
3.17 Install the ece Script	24
3.18 Assemble and Deploy	26

3.19 Verify the Installation	26
3.20 Install a Daemon Script	27
3.21 Create a Publication	28
3.22 Test Web Studio	32
3.23 Test Content Studio	33
4 Recommended Modifications	34
4.1 Configure SSL Support	34
4.1.1 Using Self-Signed Certificates	35
4.2 Search Engine Deployment and Configuration	35
4.3 Distributed Memory Cache	37
4.3.1 Install memcached	37
4.3.2 Install the memcached Java Libraries	38
4.3.3 Configure memcached	38
4.3.4 Configure PresentationArticleCache Instances	39

1 Introduction

This **Installation Guide** is intended to be read by the person responsible for

installing the Escenic Content Engine on one or more hosts. It should be read together with the **Escenic Content Engine Server Administration Guide**, which contains descriptions of the periodic administration tasks a system administrator needs to carry out once the Content Engine is installed and in operation, since there is considerable overlap between installation and maintenance of the Content Engine.

Both this manual and the **Escenic Content Engine Server Administration Guide** make the following assumptions about the Escenic installation and you, the reader:

- The Content Engine is to be installed on one or more host computers running an operating system listed in [section 1.1.2](#).
- The supporting software components (database, web server, application server and so on) listed in [section 1.1.2](#) are either already installed on these computers, or available for installation.
- You are a suitably qualified system administrator with a working knowledge of both the operating system on which the Content Engine is to be installed and of the components in the supporting software stack.

1.1 Installation Components

A working Escenic Content Engine installation depends on many components. Some of these components are delivered by Escenic as part of the product, others are third-party products on which the Content Engine depends.

1.1.1 Escenic Components

The components supplied by Escenic are:

Escenic Content Engine

The Content Engine itself, a content management system.

escenic-admin

A server administration web application.

Escenic Web Studio

A publication administration web application.

Escenic Content Studio

A Java desktop application for creating, editing and managing publication content. Although this is a desktop application and runs on end-user's PCs, it is served to users from the Content Engine via Java Webstart, and is therefore part of the Content Engine installation.

Publications

The installation includes a few small demo publications for test, demonstration and instructional purposes.

Assembly tool

The assembly tool is an ant build file that automates the application assembly and deployment process.

1.1.2 Third-Party Components

The Content Engine depends on the following third-party components:

Java Development Kit (JDK)

This provides the Java runtime and development environment required by J2EE application servers.

Database

Used to store publication content, structure and so on.

J2EE Java application server

The Content Engine is a JEE web application, and therefore runs inside a JEE application server. **escenic-admin**, Web Studio and all Escenic publications are also JEE applications and therefore also run within a JEE application server.

Web server (optional)

J2EE applications have built-in web servers that can be used for test and development purposes. For large-scale production installations, however, a separate web server may be advisable.

JDBC (Java DataBase Connectivity) driver

This is required to connect the Content Engine to the database.

Apache Ant

This is a Java-based build tool that is used by the Content Engine assembly tool.

Apache Solr

This is a Java-based search engine. It is used to provide search functionality both in web publications and in Content Studio. Unlike the other third-party components listed here, Solr is actually installed with the Content Engine.

Caching Server (optional)

A caching server such as Squid or Varnish can greatly improve the performance of heavily-loaded sites by caching frequently requested pages and sending the cached copies on request unless the original pages have been updated.

Load Balancing Server (optional)

If your site has multiple web hosts, then you may also want to install a load-balancing server to distribute requests evenly between them.

memcached (optional)

If your site has multiple web hosts, then you can improve caching efficiency by using this open source distributed cache manager to provide a shared cache for all your hosts.

In principle, the Content Engine can work with many different third-party components. In [section 2.1](#), however, you will find the list of officially supported software. The current version of the Content Engine has been tested with these components and is known to work satisfactorily.

1.2 Installation Types

The Content Engine can be installed in a number of different ways to satisfy different needs. Common installation types include:

Single computer

Everything (database, application server, Content Engine) is installed on a single computer. This is usually the case for demonstration and developer installations.

Multiple application servers, single database

The application server and Content Engine is installed on multiple computers, but all share a single database server. This kind of installation is common in large development environments.

Single web server, single application server and Content Engine, single database

This is a common production installation structure for small sites.

Multiple web servers, multiple application servers/Content Engines, database rack

This is a common production installation structure for large sites, and is easily scalable.

Other configurations are, of course, possible.

In large organizations the Content Engine often needs to be installed in several different configurations in order to meet the requirements of different environments:

Development

In a development environment there are typically several updates of several files several times a day. Depending on the number of developers working on a system and the potential for conflicting changes, developers may either:

- Share a single development installation, or
- Each use a personal Content Engine installation but share a single database.

Test

A test environment is a complete Content Engine installation on which 'current stable versions' can be tested. Code updates in this environment are typically carried out at longer intervals and in 'batches' rather than on the file level. In some organizations, the test environment may be merged with the staging environment.

Staging

A staging environment should be as similar to the production environment as practically possible: it should have the same software, the same configuration, and if possible the same hardware, firewall and network setup. This is the last stage before actual production, and is intended to catch "works on the test server but not in production" problems.

Production

A production environment is heavily optimized for high workloads and security.

1.3 Conventions Used in This Manual

This section describes various conventions used in this manual.

1.3.1 Identifying Host Machines

The Content Engine and the software it depends on may need to be installed on one or several host machines depending on the type of installation required (see [section 1.2](#)). In order to unambiguously identify the machines on which various installation actions must be carried out, the following special host names are used throughout this manual:

assembly-host

The machine used to assemble the various Content Engine components into a enterprise archive or .EAR file.

database-host

The machine used to host the database server.

engine-host

The machine(s) used to host application servers and Content Engine instances.

editorial-host

engine-host(s) that are used solely for (internal) editorial purposes. All your Content Studio clients communicate with these hosts.

presentation-host

engine-host(s) that are used solely for (public) presentation purposes. Readers' browsers communicate with these hosts.

web-host

The machine(s) used to host web servers (if needed).

share-host

The machine used to host the NFS server (if needed).

These host names always appear in a **bold typeface**. If you are installing everything on one host you can, of course, ignore them: you will be doing everything on the same machine. If you are creating a larger multi-host installation, then they should help ensure that you do things in the right places.

1.3.2 Shell Commands

All shell command examples given in the manual are tested on Debian Linux servers: they may need minor modifications to be used on other Linux or UNIX platforms, and it is assumed that you are able to make the necessary "conversions" to your own platform. Some of the commands should be executed as the owner of the Content Engine installation. This is signalled by use of the **\$** command prompt. For example:

```
| $ ls
```

Other commands must be executed as **root**. This is signalled by the use of the **#** command prompt:

```
| # /etc/init.d/slapd restart
```

Many of the code and command line examples in the manual contain placeholders: words printed in an *italic* typeface, which you are expected to replace with an appropriate value. For example:

```
| $ cat filename
```

The accompanying text usually specifies what kind of value any placeholder(s) represent: *filename* in this case is the name of a file you want to see the contents of.

1.3.3 Standard File Hierarchy

All file paths and URLs shown in the manual are based on the following standard folder structure:

Standard location	Component
<code>/opt/escenic</code>	Escenic
<code>/opt/escenic/engine</code>	Escenic Content Engine
<code>/opt/escenic/assemblytool</code>	Escenic assembly tool
<code>/etc/escenic</code>	Escenic configuration
<code>/etc/escenic/engine</code>	Escenic Content Engine configuration
<code>/opt/java/jdk</code>	Java
<code>/opt/java/ant</code>	Ant
<code>/opt/tomcat</code>	Tomcat

If your system is organized differently, then adjust the paths you use accordingly.

2 Product Requirements

The Escenic Content Engine is written in Java, runs on an application server, and uses a database to store content and user data. One of each of these supporting products must be installed before the Content Engine can be installed. The specific supporting products with which the Content Engine is known to work are listed in [section 2.1](#). Note, however, that these products have their own hardware and software requirements. Please make sure that your environment meets the requirements stipulated by the relevant product suppliers.

Some additional, more general requirements and recommendations are listed in [section 2.2](#).

2.1 Required Supporting Software

The Content Engine requires a combination of the following software components:

Operating System

- Ubuntu Server (Long Term Support version recommended). See the **Installation Guide for Linux** if you are installing on this platform.
- Solaris 10. See the **Installation Guide for Linux** if you are installing on this platform.

Application Server

- Apache Tomcat 7.0

Database Server

- MySQL 5.0.x
- Oracle Database 10.1 or later

Java SE Development Kit

Oracle Java SE Development Kit 7

Web Server

Apache Web Server Version 2.2.12

2.2 Other Requirements

2.2.1 Shared Filesystem

In large installations, the web server may well run on a different host to the application server. However, the web server and application server need access to a common file system, since the Content Engine writes and updates files that are served by the web server (multimedia files, for example). On Linux systems, NFS can be used to set up the required shared folders. For Further information, see [section 3.5](#).

2.2.2 SSH/FTP Server

The Content Engine does not need an SSH or FTP Server. Setting up an SSH or FTP server is, however, highly recommended. An SSH/FTP Server can be used by template developers, among others, to upload new and updated files to the server. If it is not possible to install an SSH or FTP server on the **engine-host** (for security reasons, perhaps), then some other way of uploading files must be provided.

2.2.3 Apache Ant

Ant is used to package an Enterprise Archive (EAR file) containing the Content Engine and all the web applications required to run it.

Ant is only required on the **assembly-host**. It is not necessary to install ant on all of the hosts in a cluster.

3 Installation Procedure

This chapter contains step-by-step instructions for installing the Content Engine on a single host computer or on a cluster of several host computers. The instructions use the host names listed in [section 1.3.1](#) to indicate where you should carry out various steps. If some of your hosts are "multi-purpose" (if for example, your database server is installed on the same host as your editorial Content Engine installation), then carry out all appropriate steps on that machine (for example, all **database-host** steps and all **editorial-host** steps).

Some of the steps are not required for single-host deployment. These steps are clearly marked.

The Content Engine can be installed in many different ways. You are, however, recommended to follow the procedure described below as it will result in a standardized, easily-maintained installation structure.

3.1 Install Java SE Development Kit (JDK)

On your **assembly-host** and **engine-host(s)**, while logged in as **root**, download and install Java SE development kit (JDK), version 7.

You can download the JDK from <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Follow the installation instructions supplied on the Oracle site. When you have installed it you need to make sure that it is installed as the default Java installation. You can do this by entering the following commands while logged in as **root**:

```
# update-alternatives --install "/usr/bin/java" "java" "java-install-path" 1
# update-alternatives --set java java-install-path
```

where *java-install-path* is the path of the location in which you installed Java (for example, **/usr/lib/jvm/jdk1.7.0_11/jre/bin/java**).

Verify that Java is correctly installed by entering:

```
# java -version
```

This should result in output something like this:

```
java version "1.7.0_11"
Java(TM) SE Runtime Environment (build 1.7.0_11-b21)
Java HotSpot(TM) 64-Bit Server VM (build 23.6-b04, mixed mode)
```

3.2 Install Ant

On your **assembly-host**, while logged in as **root**:

Download a binary distribution of **ant** from <http://ant.apache.org/bindownload.cgi>. Unpack the distribution to an appropriate place and follow the installation instructions given here:

<http://ant.apache.org/manual/install.html#installing>.

It may also be possible to download **ant** from one of your Linux distribution's software repositories. On Debian-based Linux machines, for example, you can install it by entering:

```
| # apt-get install ant
```

Verify that **ant** is correctly installed by entering:

```
| # ant -version
```

This should produce a response something like this:

```
| Apache Ant version 1.7.1 compiled on September 8 2010
```

3.3 Install Various Utilities

Make sure the following utilities are available on all your hosts:

- **unzip**
- **telnet**

On Debian-based Linux machines, you can easily install these by entering the following commands:

```
| # apt-get install unzip  
| # apt-get install telnet
```

You may also find it useful to install an SSH server on all your hosts so that you can easily switch sessions and copy files between them. On Debian-based Linux machines, you can do this as follows:

```
| # apt-get install openssh-server
```

3.4 Create Escenic Users

All the Content Engine components should run under the same user, and this user should have the same UID on all hosts. So create a user (preferably called **escenic**) on each host machine as follows:

```
| # adduser scenic
```

and make sure they all have the same UID. If you are installing on clean machines, and **escenic** is the first user you create, then they will all have the same UID by default.

On your **assembly-host** and **engine-host(s)**, make sure that Java is in the path of the **escenic** users:

```
| # su - scenic  
$ echo 'export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_11' >> .bashrc  
$ echo 'export PATH=$JAVA_HOME/bin:$PATH' >> .bashrc  
$ source ~/.bashrc
```

3.5 Create Shared File System

If you are installing everything on a single host, you can skip this section.

If you are installing the Content Engine on multiple hosts then it is usually a good idea to create some shared folders that can be accessed from all the hosts. If your **engine-host** is not the same machine as your **web-host**, then it is more or less a requirement, since the Content Engine creates and modifies multimedia files that need to be accessed by the web server. (If you cannot create shared folders for some reason, then you need to find some other means of synchronizing files between the hosts.)

Shared folders can also be useful for other reasons, such as access to downloads or shared configuration files, and these instructions assume that you will use shared folders for these purposes.

The following instructions show how to set up three shared folders: **ece-conf**, **multimedia** and **download**.

On your **share-host**, while logged in as **root**:

1. Download and install the NFS server software. For example, on a Debian-based Linux distribution:

```
# apt-get install nfs-kernel-server
```

2. Create the folders to be shared and set **escenic** as their owner:

```
# mkdir -p /exports/ece-conf
# mkdir -p /exports/multimedia
# mkdir -p /exports/download
# chown escenic:escenic /exports/ece-conf /exports/multimedia /exports/download
```

3. Open the NFS exports file **/etc/exports** and add an entry for each folder that you want to share:

```
/exports/ece-conf    subnet-specification(rw,sync)
/exports/multimedia  subnet-specification(rw,sync)
/exports/download    subnet-specification(rw,sync)
```

where *subnet-specification* identifies the IP address range within which the shared folders are to be accessible. If, for example, all your hosts have IP addresses in the **192.168.71.nnn** subnet, then you might enter the subnet specification **192.168.71.0/255.255.255.0**.

4. Restart all NFS-related services:

```
# /etc/init.d/portmap restart
# /etc/init.d/nfs-kernel-server restart
# /etc/init.d/nfs-common restart
```

5. Create symbolic links to the shared folders as follows:

```
# ln -s /exports/ece-conf /mnt/ece-conf
# ln -s /exports/multimedia /mnt/multimedia
# ln -s /exports/download /mnt/download
```

This ensures that the shared folders will be accessible via the same paths on your **share-host** as on all the other hosts.

6. Change the owner of the links to **escenic**:

```
# chown escenic:escenic /mnt/ece-conf
# chown escenic:escenic /mnt/multimedia
# chown escenic:escenic /mnt/download
```

On all your other hosts, while logged in as **root**:

1. Download and install the NFS client software. For example, on a Debian-based Linux distribution:

```
# apt-get install nfs-common
```

2. Create mount points for the shared folders:

```
# mkdir /mnt/ece-conf
# mkdir /mnt/multimedia
# mkdir /mnt/download
```

3. Open **/etc/fstab** and add an entry for each shared folder:

```
share-host:/exports/ece-conf /mnt/ece-conf nfs defaults 0 0
share-host:/exports/multimedia /mnt/multimedia nfs defaults 0 0
share-host:/exports/download /mnt/download nfs defaults 0 0
```

where *share-host* is the host name or IP address of your **share-host**.

4. Mount the shared folders:

```
# mount -a
```

3.6 Download Content Engine

You will need login credentials to be able to download Content Engine and other components that you need. If you have not already received a username and password for this purpose, please contact your Escenic representative or Escenic support (support@escenic.com).

On your **assembly-host**, while logged in as **escenic**:

Download the following items from <http://documentation.vizrt.com/ece-download-matrix/5.6/>:

- The Content Engine installation set **engine-5.6.13.183224.zip**
- The Content Engine assembly tool **assemblytool-2.0.5.zip**
- The Content Engine documentation package **documentation-engine-5.6.13.183224-pdf.zip** (if you do not already have a full, up-to-date documentation set)

If you are installing on more than one machine and have created the shared folders described in [section 3.5](#), then it is a good idea to download these packages to the **/mnt/download** folder you created. Otherwise, download them to some temporary location of your choice.

You can download the files by entering the following commands:

```
$ cd /mnt/download
$ wget http://user:password@technet.escenic.com/downloads/release/56/
engine-5.6.13.183224.zip
$ wget http://user:password@technet.escenic.com/downloads/assemblytool-at-version.zip
$ wget http://user:password@technet.escenic.com/downloads/release/56/documentation-
engine-5.6.13.183224-pdf.zip
```

where:

- *user* and *password* are the user name(s) and password(s) you have received from Escenic
- *at-version* is the number of the latest version of the assembly tool (2.0.5 or later)

3.7 Install Database

The following instructions describe how to install and set up MySQL for use by the Content Engine.

On your **database-host**, while logged in as **root**:

1. Install the MySQL server and client packages. For example, on a Debian-based Linux distribution:

```
# apt-get install mysql-server mysql-client
```

During the installation you will be asked to specify a root password for the database.

2. Log in to the system and create a database for the Content Engine:

```
# mysql -p
mysql> create database db-name character set utf8 collate utf8_general_ci;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on db-name.* to user@'%' identified by 'password';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all on db-name.* to user@'localhost' identified by 'password';
Query OK, 0 rows affected (0.00 sec)
mysql> exit
```

Replace *db-name*, *user* and *password* in the above commands with a database name, user name and password of your choice.

3. If you are installing everything on a single host or if you created a shared file system as described in [section 3.5](#), then you should have direct access to the Content Engine package you downloaded earlier (see [section 3.6](#)). If not, you will need to copy it to some temporary location on the **database-host**.

4. Change user to **escenic** and unpack the Content Engine package to a temporary location:

```
# su - scenic
$ cd /tmp
$ unzip /mnt/download/engine-5.6.13.183224.zip
```

5. Still as the **escenic** user, run the Content Engine's database scripts:

```
$ cd /tmp/engine-5.6.13.183224/database/mysql/
$ for el in tables.sql indexes.sql constants.sql constraints.sql; do \
    mysql -u user -ppassword db-name < $el
done;
```

Replace *db-name*, *user* and *password* in the above commands with the names you chose in step 2. Your *password* must **immediately follow** the **-p** switch, with no intervening space (as shown above).

6. On some platforms, external access to the MySQL server is disabled by default. To enable external access, you need to bind the **mysql** process to the **database-host**'s IP address. To do this, you need to open **/etc/mysql/my.cnf** for editing (as **root** again) and set the **bind-address** parameter:

```
bind-address = database-host-ip-address
```

You should then verify that the MySQL server is running and accessible by trying to connect to port 3306 from each of your other hosts using **telnet**:

```
$ telnet database-host 3306
```

where *database-host* is the host name or IP address of the **database-host**. If a connection is opened, then the database server is running and accessible.

On a single-host installation, you can check that MySQL is running by entering the following command:

```
$ mysqladmin -u root -p status
```

You should get a response something like this:

```
Uptime: 605 Threads: 1 Questions: 615 Slow queries: 0 Opens: 842 Flush tables: 1 Open
tables: 40 Queries per second avg: 1.16
```

If the server is not running then you will see an error reporting that it was not possible to connect to the server.

3.8 Install Application Server

The following instructions describe how to install and set up the Apache Tomcat application server for use by the Content Engine.

On your **engine-host(s)**, while logged in as **escenic**:

1. Download the latest Tomcat package from <http://tomcat.apache.org/>. If you are installing on more than one machine and have created the shared folders described in [section 3.5](#), then it is a good idea to download these packages to the **/mnt/download** folder you created. Otherwise, download them to some temporary location of your choice.

2. Change user to **root** and unpack the Tomcat package to **/opt**:

```
$ su
# cd /opt
# tar -zxvf /mnt/download/apache-tomcat-version-number.tar.gz
```

where *version-number* is the version number of the package you have downloaded.

3. Create a symbolic link from **/opt/tomcat** to the Tomcat folder, so that it will be easier to upgrade to new versions when necessary, and change the owner of the Tomcat folder to **escenic**.

```
# ln -s /opt/apache-tomcat-version-number /opt/tomcat
# chown -R scenic:escenic /opt/apache-tomcat-version-number
```

4. Open Tomcat's configuration file file (**/opt/tomcat/conf/catalina.properties**) for editing and add the following characters:

```
,${catalina.home}/escenic/lib/*.jar
```

to the end of the **common.loader** property setting.

5. Download a JDBC driver for the database system you are using. For MySQL, the driver is called **Connector/J** and can be downloaded from <http://dev.mysql.com/downloads/connector/>.
6. Change user back to **escenic** and install the driver by copying it to the **/opt/tomcat/lib** folder. If you downloaded the MySQL driver package to **/mnt/download**, then you can do this as follows:

```
# su - scenic
```



```
$ cd /tmp
$ tar -zxvf /mnt/download/mysql-connector-java-version-number.tar.gz
$ cp mysql-connector-java-version-number/mysql-connector-java-version-number-
bin.jar \
> /opt/tomcat/lib/
```

7. Create the **/opt/tomcat/escenic/lib** folder you added to the **common.loader** path in step 4:

```
$ mkdir -p /opt/tomcat/escenic/lib/
```

8. Set up database pooling. For MySQL, you do this by opening **/opt/tomcat/conf/context.xml** for editing and inserting the following two resource definitions as children of the root **Context** element:

```
<Resource
  name="jdbc/ECE_READ_DS"
  auth="Container"
  type="javax.sql.DataSource"
  username="user"
  password="password"
  driverClassName="com.mysql.jdbc.Driver"
  maxActive="30"
  maxIdle="10"
  maxWait="5000"
  url="jdbc:mysql://database-host/db-name?
autoReconnect=true&useUnicode=true&characterEncoding=UTF-8"
/>

<Resource
  name="jdbc/ECE_UPDATE_DS"
  auth="Container"
  type="javax.sql.DataSource"
  username="user"
  password="password"
  driverClassName="com.mysql.jdbc.Driver"
  maxActive="10"
  maxIdle="5"
  maxWait="5000"
  url="jdbc:mysql://database-host/db-name?
autoReconnect=true&useUnicode=true&characterEncoding=UTF-8"
/>
```

Replace *database-host*, *db-name*, *user* and *password* in the above definitions with the names you have defined earlier (see [section 3.7](#)).

If you copy and paste the above resource definitions, make sure to remove the line breaks in the **url** values (caused by page width restrictions).

9. Set up indexing by inserting the following elements in **/opt/tomcat/conf/context.xml** as children of the root **Context** element:

```
<Environment
  name="escenic/indexer-webservice"
  value="http://indexer-web-service-host:8080/indexer-webservice/web-service-
name/"
  type="java.lang.String"
  override="false"/>

<Environment
  name="escenic/index-update-uri"
  value="http://localhost:8080/solr/update/"
```

```

        type="java.lang.String"
        override="false"/>

<Environment
    name="escenic/solr-base-uri"
    value="http://localhost:8080/solr/"
    type="java.lang.String"
    override="false"/>

<Environment
    name="escenic/head-tail-storage-file"
    value="/var/lib/escenic/head-tail.index"
    type="java.lang.String"
    override="false"/>

<Environment
    name="escenic/failing-documents-storage-file"
    value="/var/lib/escenic/failures.index"
    type="java.lang.String"
    override="false"/>

```

where:

- On an **editorial-host**, *indexer-web-service-host* is the host name or IP address of the **editorial-host** on which the Content Engine's internal indexer web service is to run and *web-service-name* is **index**.
- On a **presentation-host**, *indexer-web-service-host* is the host name or IP address of the **presentation-host** on which the Content Engine's external indexer web service is to run and *web-service-name* is **presentation-index**.

If you are creating a single-host installation, then you can use the host name **localhost** and the indexer web service name **index**.

10. Open Tomcat's **web.xml** file (**/opt/tomcat/conf/web.xml**) for editing and insert links to the resources you have defined:

```

<resource-ref>
  <description>Read link</description>
  <res-ref-name>jdbc/ECE_READ_DS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>

<resource-ref>
  <description>Update link</description>
  <res-ref-name>jdbc/ECE_UPDATE_DS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>

```

These elements must be inserted as children of the root **web-app** element: otherwise, position is irrelevant.

11. Open **/opt/tomcat/conf/server.xml** for editing. Somewhere in this file you will find a **Connector** element that configures connections on port 8080. Make sure that this element contains a **URIEncoding** attribute, and that it is set to **UTF-8**. For example:

```

<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443"

```

```
|      URLEncoding="UTF-8"/>
```

This ensures that Content Engine search functionality works for non-Latin characters.

3.9 Create Configuration Folder

While logged in as **root**, create an **escenic** folder under **/etc**, and change its owner to **escenic**:

```
| # mkdir /etc/escenic
| # chown escenic:escenic /etc/escenic
```

If you are installing everything on one host, then:

While logged in as **escenic**, create an **engine** folder under **/etc/escenic**:

```
| $ mkdir /etc/escenic/engine
```

If you are creating a multi-host installation, then:

On one of your **engine-hosts**, while logged in as **escenic**, create a symbolic link from **/etc/escenic/engine** to **/mnt/ece-conf**:

```
| $ ln -s /mnt/ece-conf /etc/escenic/engine
```

3.10 Unpack Content Engine Components

If you are installing everything on a single host or if you created a shared file system as described in [section 3.5](#), then you should have direct access to the Content Engine packages you downloaded earlier (see [section 3.6](#)). If not, you will need to copy it to some temporary location on the **assembly-host**. The following instructions assume that you have downloaded the packages to the shared folder **/mnt/download**.

On your **assembly-host**, while logged in as **root**:

1. Create a folder for the Content Engine under **/opt**:

```
| # mkdir /opt/escenic
| # chown -R escenic:escenic /opt/escenic
```

2. Change user to **escenic** and unpack the Content Engine package as follows:

```
| # su - escenic
| $ cd /opt/escenic/
| $ unzip /mnt/download/engine-5.6.13.183224.zip
```

3. Create a symbolic link from **/opt/escenic/engine** to the Content Engine folder, so that it will be easier to upgrade to new versions when necessary.

```
| $ ln -s engine-5.6.13.183224 engine
```

4. Create a folder and symbolic link for the assembly tool, then unpack the assembly tool into the folder:

```
| $ mkdir assemblytool-2.0.5
| $ ln -s assemblytool-2.0.5 assemblytool
| $ cd assemblytool/
```

```
$ jar xf /mnt/download/assemblytool-2.0.5.zip
```

Note that the **jar** command shown above is part of the JDK, and is located in the **bin** directory of the Java distribution. Some Linux distributions ship with a GNU **jar** command which has a different syntax and does not support all the same options. Make sure you use the JDK's **jar** command.

3.11 Logging

On one of your **engine-hosts**, while logged in as **escenic**:

1. Create the folder **/etc/escenic/engine/common**:

```
$ mkdir /etc/escenic/engine/common
```

2. Create a logging configuration file called **trace.properties** in the new folder. This file should have the following content:

```
log4j.rootCategory=ERROR
log4j.category.com.escenic=ERROR, ECELOG
log4j.category.neo=ERROR, ECELOG

log4j.appender.ECELOG=org.apache.log4j.DailyRollingFileAppender
log4j.appender.ECELOG.File=/var/log/escenic/engine/ece-messages.log
log4j.appender.ECELOG.layout=org.apache.log4j.PatternLayout
log4j.appender.ECELOG.layout.ConversionPattern=%d %5p [%t] %x (%c) %m%n
```

On all your **engine-hosts**, while logged in as **escenic**:

1. Create the output folder specified in **trace.properties**:

```
$ mkdir /var/log/escenic/engine
```

2. In order to make Tomcat use the **trace.properties** file, create a link to it from **/opt/tomcat/lib**:

```
$ cd /opt/tomcat/lib
$ ln -s /etc/escenic/engine/common/trace.properties
```

3. In order to make Tomcat output its log files to a standard Unix/Linux location, open **/opt/tomcat/conf/logging.properties** for editing and set **all** the log folder properties (they have names ending with **FileHandler.directory**) to **/var/log/tomcat**.

3.12 Copy Solr Configuration

The Solr configuration data supplied with the Content Engine installation needs to be copied to locations where it can be accessed by all the Solr instances running in your system. By default, this means that you will need a copy of the Solr configuration data in the **/var/lib/escenic/solr/default** folder on each of your **engine hosts**.

To achieve this on a single host installation, log in as **escenic** and enter the following commands:

```
$ mkdir -p /var/lib/escenic/solr
$ cp -r /opt/escenic/engine/solr/* /var/lib/escenic/solr/
```

On a multiple host installation, log in as **escenic** on your **assembly host** and copy the Solr configuration data to the required destination folder on each **engine host**. For example, enter the following commands for each **engine host**:

```
$ ssh engine-host "mkdir -p /var/lib/escenic/solr"
$ scp -r /opt/escenic/engine/solr/* engine-host:/var/lib/escenic/solr
```

3.13 Initialize the Assembly Tool

On your **assembly-host**, while logged in as **escenic**:

1. **cd** to the **/opt/escenic/assemblytool** folder:

```
$ cd /opt/escenic/assemblytool
```

2. Enter the following command:

```
$ ant initialize
```

This command creates the **.properties** files the assembly tool needs.

3. Open **/opt/escenic/assemblytool/assemble.properties** for editing.
4. Uncomment the **engine.root** setting near the top of the file, and set it as follows:

```
engine.root = /opt/escenic/engine
```

3.14 Initialize the Bootstrap Layer

On your **assembly-host**, while logged in as **escenic**:

1. **cd** to the **/opt/escenic/assemblytool** folder:

```
$ cd /opt/escenic/assemblytool
```

2. Enter the following command:

```
$ ant -q ear
```

This command (which takes a few minutes to complete) creates the **bootstrap layer**. The bootstrap layer is a set of configuration files that "configure the configuration process". Once you have run the above command, you will find the bootstrap layer located in **/opt/escenic/assemblytool/conf**. This bootstrap layer is set up to look for three configuration layers. The first two layers (**default** and **addon**) are part of the delivered system, and do not concern you. The third layer is called **common**, and the bootstrap layer is set up to look for it in **/etc/escenic/engine/common**. You will therefore need to create a common configuration layer there, as described in [section 3.15](#).

3.15 Create the Common Configuration Layer

The Content Engine package installed in the [section 3.10](#) step contains a **skeleton configuration layer**. This is a folder tree containing configuration files that you can use as a basis for your configurations. All the property settings in these files are commented out. To create a configuration, you copy this tree to the location specified in your bootstrap layer (e.g. **/etc/escenic/engine/**

common) and edit the contents to set the properties you need. You will find the skeleton configuration layer located in **/opt/escenic/engine/siteconfig/config-skeleton**.

On your **assembly-host**, while logged in as **escenic**:

1. Copy the skeleton configuration layer to **/etc/escenic/engine/common**:

```
$ cp -r /opt/escenic/engine/siteconfig/config-skeleton/* /etc/escenic/engine/
common/
```

2. Copy the delivered security configuration files to **/etc/escenic/engine/common**

```
$ cp -r /opt/escenic/engine/security/ /etc/escenic/engine/common/
```

3. Open **/etc/escenic/engine/common/ServerConfig.properties** for editing, and make sure the following properties are present, uncommented and set correctly for your site:

databaseProductName

The name of the database you are using. Currently allowed values are:

- **MySQL**

filePublicationRoot

The path of the folder in which all binary files served by the Content Engine are stored (multimedia files - video, audio, images, Word documents, PDF files and so on). You should set this to:

```
filePublicationRoot=/var/lib/escenic/publications/
```

The path you specify **must** include a trailing slash.

webPublicationRoot

The URL root you want to be used for all your publications. For example **http://my-company.com/**. The URLs of your publications are formed by appending the publication name to this string. This property is in general used only for development purposes. In production, it is often the case that each publication has its own root URL.

customerId

The name of your organization, or some other name that will clearly identify your installation to Escenic support staff. Reporter components that are installed with the Content Engine automatically send reports about the Escenic software running at your site. The reported information includes:

- Which versions of the Content Engine and its plug-ins are running at your site.
- The number of active Content Studio users (reported once an hour)

This information is very valuable to Escenic support engineers and helps with the diagnosis of reported errors and performance problems.

The reporter components sends their reports to Google Analytics. In order for this to work, your firewalls must allow outgoing traffic on port 80 to **http://www.google-analytics.com**.

4. Open **/etc/escenic/engine/common/connector/ReadConnector.properties** for editing, and make sure the following property is uncommented and set correctly:

dataSourceName

Must be set as follows:

```
dataSourceName=java:comp/env/jdbc/ECE_READ_DS
```

This is the name of the read database pool that you defined when installing the application server (see [section 3.8](#)).

Do not modify any of the other settings in this file.

5. Open `/etc/escenic/engine/common/connector/UpdateConnector.properties` for editing, and make sure the following property is uncommented and set correctly:

dataSourceName

Must be set as follows:

```
dataSourceName=java:comp/env/jdbc/ECE_UPDATE_DS
```

This is the name of the update database pool that you defined when installing the application server (see [section 3.8](#)).

Do not modify any of the other settings in this file.

6. Open `/etc/escenic/engine/common/neo/io/managers/ContentManager.properties` for editing, and make sure the following properties are uncommented and set correctly:

readConnector

Must be set as follows:

```
readConnector=/connector/ReadConnector
```

updateConnector

Must be set as follows:

```
updateConnector=/connector/UpdateConnector
```

3.16 Create Host Configuration Layers

If you are installing everything on one host, then skip this section.

In a multi-host installation, a small number of configuration properties may need to be set differently on different hosts. This can be achieved by creating individual configuration layers for each host. The Content Engine instance on each host reads its host configuration layer **after** the common configuration layer, and any settings it finds there override both system defaults and any values set in the common configuration layer.

On your **assembly-host**, while logged in as **escenic**, create a folder called `/etc/escenic/engine/host`. Under this folder, create a folder for each machine that is to host a Content Engine instance. If, for example, your installation has one **editorial host** (called **editorial1**) and one **presentation host** (called **presentation1**), then you should enter:

```
$ mkdir /etc/escenic/engine/host/
$ mkdir /etc/escenic/engine/host/editorial1/
$ mkdir /etc/escenic/engine/host/presentation1/
```

If there are any other configuration settings that you know need to be set differently on a specific host, you can set them in the same way:

1. Copy the appropriate **.properties** file from the skeleton configuration layer to the same relative location in the appropriate host configuration layer.

2. Open the copied file for editing.
3. Uncomment the required property and set it to the required value.

For further information about the configuration layers, see **Escenic Content Engine Server Administration Guide, chapter 4**.

3.17 Install the ece Script

The **ece** shell script provides a set of commands for easily stopping, starting and assembling the Content Engine.

To install the **ece** shell script:

On your **assembly-host**, while logged in as **escenic**:

1. Make **/opt/escenic/engine/ece-scripts/usr/bin/ece** executable:

```
$ chmod +x /opt/escenic/engine/ece-scripts/usr/bin/ece
```
2. Open **/home/escenic/.bashrc** for editing, and add the following line to it in order to add **/opt/escenic/engine/ece-scripts/usr/bin** to your **PATH**:

```
export PATH=/opt/escenic/engine/ece-scripts/usr/bin:$PATH
```
3. Then enter the following command to apply the change you have made in your current shell:

```
$ source ~/.bashrc
```
4. Copy the script's configuration file to **/etc/escenic/engine**:

```
$ cp /opt/escenic/engine/ece-scripts/etc/escenic/ece.conf /etc/escenic/
```

This is the configuration file that will be read when **ece** is executed on the **assembly-host**.

5. If you are installing everything on one host, then skip this step.

Copy the script's configuration file to each of the host configuration folders you have made. For example:

```
$ cp /opt/escenic/engine/ece-scripts/etc/escenic/ece.conf /etc/escenic/ece-  
editorial1.conf  
$ cp /opt/escenic/engine/ece-scripts/etc/escenic/ece.conf /etc/  
escenic/ece-presentation1.conf
```

These are the configuration files that will be read when **ece** is executed on the various **engine-hosts**. (If your **assembly-host** is also an **engine-host**, then you do not need a copy in **/etc/escenic/engine**, the host-specific copy will be used.)

6. Open the copies of **ece.conf** that you created in steps 4 (and possibly 5), and make sure that the parameters listed below are set correctly for your installation, and for the specific host they correspond to. If you have followed all the instructions in this guide exactly, then the following settings should work:

java_home

Make sure that this is set to the path of the Java virtual machine included in the JDK you installed (see [section 3.1](#)).

ece_home

Set this to **/opt/escenic/engine**.

escenic.server

Set this to the host name or IP address, should be unique for each server in a multi-server setup (not relevant for single-host installations).

ece_security_configuration_dir

Set this to `/etc/escenic/engine/common/security`.

appserver

Currently, the only valid setting for this is `tomcat`.

tomcat_home

Set this to `/opt/tomcat`.

solr_home

Set this to `/var/lib/escenic/solr`. This is the location to which you have copied the Solr configuration data (see [section 3.12](#)).

assemblytool_home

Set this to `/opt/escenic/assemblytool`.

7. Create the standard installation directory structure. Enter the following commands:

```
# mkdir -p /var/{crash,lib,log,run,cache,spool}/escenic
# chown escenic:escenic /var/{crash,lib,log,run,cache,spool}/escenic -R
```

These commands will create the following folders and assign them to the **escenic** user:

```
/var/crash/escenic
/var/lib/escenic
/var/log/escenic
/var/run/escenic
/var/cache/escenic
/var/spool/escenic
```

If, for any reason, you do not want to install Content Engine files in standard locations and you have modified any of the following settings in the `ece.conf` file, then you have to make sure that all of those exist and the **escenic** user has write permission. You are strongly advised, however, not to do so. All instructions in this manual and the **Escenic Content Engine Server Administration Guide** assume files are installed in standard locations. The configurations are:

```
| cache_dir
| log_dir
| pid_dir
| heap_dump_dir
```

Then, on each of your **engine-hosts**, while logged in as **root**:

1. Copy the **ece** script from the **assembly-host**. If you have installed an SSH server on the **assembly-host** and SSH clients on your **engine-hosts**, then you can do this as follows:

```
# scp escenic@assembly-host-ip-address:/opt/escenic/engine/ece-scripts/usr/bin/
ece /usr/bin/
```

where *assembly-host-ip-address* is the host name or IP address of your **assembly-host**.

2. Make **/usr/bin/ece** executable:

```
# chmod +x /usr/bin/ece
```

3. Now you need to make sure that the required directory structure exists. Execute the same commands to you have used to create the directories in **assembly-host** with appropriate permission.
4. Verify that the script is correctly installed by entering the following while logged in as **escenic**:

```
$ ece help
```

This should give help output describing the usage of the **ece** command.

3.18 Assemble and Deploy

Assembly and deployment consists of the following steps:

1. On your **assembly-host**, while logged in as **escenic**, run the **ece** script to assemble a set of Content Engine applications:

```
$ ece assemble
```

This generates an **enterprise archive** (EAR file) which you can deploy on all your **engine-hosts**. The EAR file is called **engine.ear**, and is created in the **/var/cache/escenic/** folder.

2. If you are installing everything on one host, then skip this step.

On each **engine-host**, copy **/var/cache/escenic/engine.ear** from the **assembly-host**. If you have installed an SSH server on the **assembly-host** and SSH clients on your **engine-hosts**, then you can do this as follows:

```
$ scp -r scenic@assembly-host-ip-address:/var/cache/escenic/engine.ear /var/cache/escenic/
```

where *assembly-host-ip-address* is the host name or IP address of your **assembly-host**.

3. On each **engine-host**, deploy the EAR file by entering:

```
$ ece deploy
```

3.19 Verify the Installation

It should now be possible to start the Content Engine and verify that the major components are working. To do so, carry out the following procedure on each **engine-host**, while logged in as **escenic**:

1. Start the Content Engine by entering:

```
$ ece start
```

2. Verify that the Content Engine is running by entering:

```
$ ece status
```

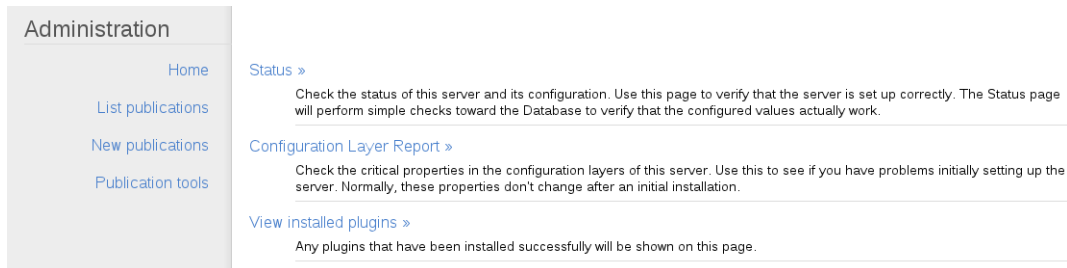
This should produce something like the following output:

```
[ece] Escenic Content Engine IS running (PID 1936) up 0d 1h 33m 51s
```

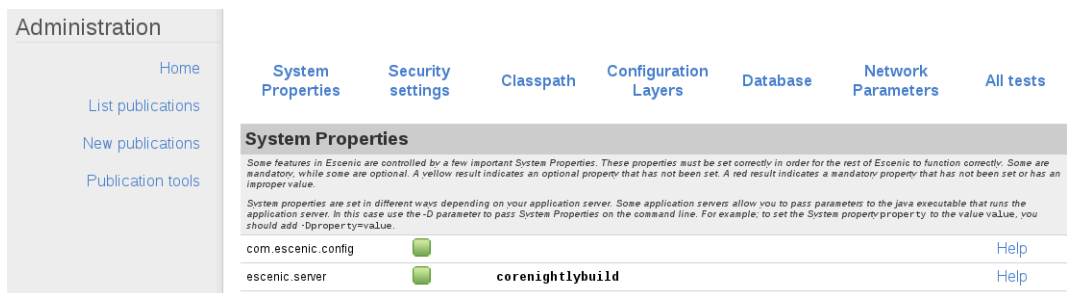
- Open a browser, and try accessing the **escenic-admin** web application on each of the **engine-hosts** by entering:

```
http://engine-host-ip-address:8080/escenic-admin/
```

in the browser address field. There may be a delay while the Content Engine initializes, but eventually, you should see the following page:



- Click on the **Status** link to display a set of status pages containing Content Engine test results.



- Across the top of these pages is a menu containing links to each of the status pages. Click on each of these links in turn and verify that all the tests pass (indicated by a icon).
- The icon indicates that something is not correctly configured. If you see one of these icons, then you need to check over all relevant settings. In most cases there is a help link next to the test result indicating the most likely causes of failure. Make any necessary corrections, then restart the Content Engine by entering:

```
$ ece restart
```

- Wait a minute or so, then refresh the browser window to re-execute the tests that failed.

Verify that the Content Engine is correctly configured on **all** your **engine-hosts** before proceeding any further.

On some of the test pages you will see warning icons. In most cases these are simply an indication that no publications have yet been created, and they can be ignored.

3.20 Install a Daemon Script

At this point you have successfully installed the Content Engine and managed its instances using the **ece** script. Now you may want to install a daemon script that will automatically run the Content Engine on system start-up and shut it down on system shutdown. If you have the database server on the same machine then you have to make sure that it also starts automatically on system start-up.

The following instructions tell you how to install the daemon script.

Log in as **root** on your **assembly-host** and on each of your **engine-hosts** and do the following:

1. Copy the daemon script **ece** to **/etc/init.d/**, and the script's default settings (**ece-scripts/etc/default/ece**) to a file called **ece** in the **/etc/default** folder. On the **assembly-host** you can do this as follows:

```
# cp /opt/escenic/engine/ece-scripts/etc/init.d/ece /etc/init.d/
# cp /opt/escenic/engine/ece-scripts/etc/default/ece /etc/default/
```

On the engine-hosts, however, you will have to copy the files from the **assembly-host**. If you have installed an SSH server on the **assembly-host** and SSH clients on your **engine-hosts**, then you can do it as follows:

```
# scp escenic@assembly-host-ip-address:/opt/escenic/engine/ece-scripts/etc/init.d/
ece /etc/init.d/
# scp escenic@assembly-host-ip-address:/opt/escenic/engine/ece-
scripts/etc/default/ece /etc/default/
```

2. Make the daemon script executable:
3. The next step is to run the daemon script, **ece**. But before that you may need to modify the settings for the daemon script. Open **/etc/default/ece** in an editor and modify the settings to meet your requirements.

If you have not installed the Content Engine files in standard locations, then you must modify the **dir_list** setting according to the **/etc/escenic/ece.conf**, i.e. the directories you have created while installing the **ece** script. See [section 3.17](#).

Now, if you have started the Content Engine, stop it.

```
# ece stop
```

Then enter the following commands:

```
# sudo /etc/init.d/ece start
```

If everything is ok at this point, then you are ready to add this script as a daemon. Other wise you need to fix the problem and then try again.

4. On your **engine-hosts** only, enter the following commands:

```
# update-rc.d ece defaults
```

This command actually installs the script as a daemon, ensuring that it will be run on system startup. Note that this command is specific to Debian-based Linux distributions: the procedure for installing daemon scripts is different on other distributions.

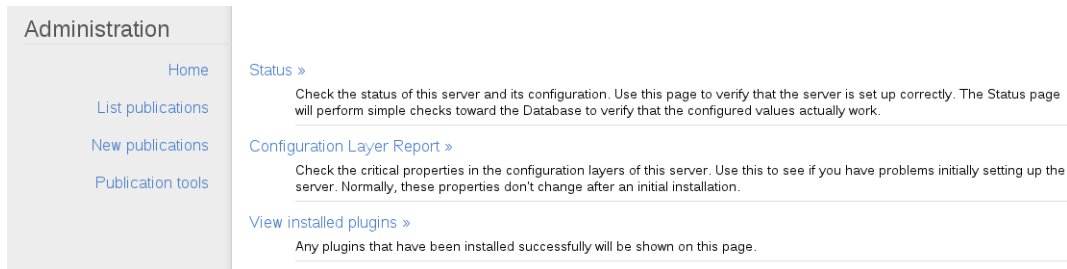
3.21 Create a Publication

To really verify that everything is working correctly, you need a publication in the database. The quickest way to achieve this is to add one of the demo publications included with the installation. To add the **demo-temp-dev** publication (which is a very simple, stripped down publication with almost no content):

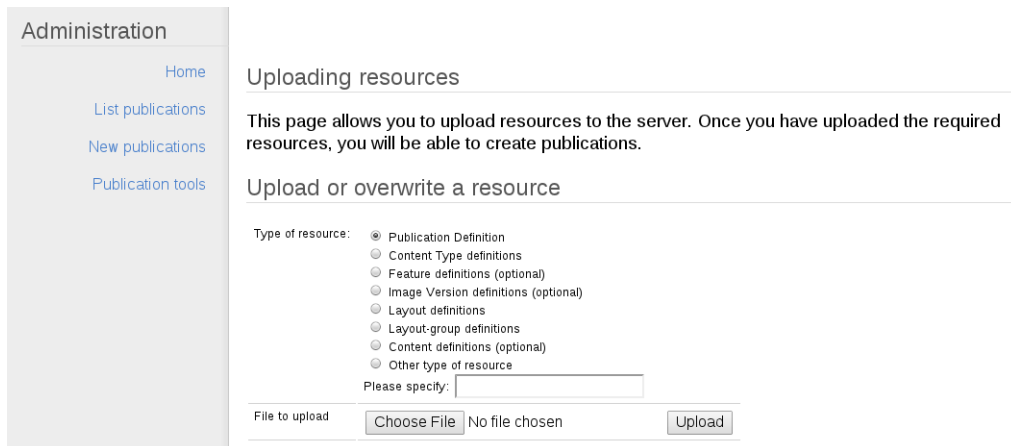
1. Download the publication WAR file from one of your **engine-hosts** to the machine on which you are working. You will find **demo-temp-dev.war** in the **/opt/escenic/engine/contrib/wars** folder.
2. Open a browser, and access the **escenic-admin** web application on one of your **editorial-hosts** by entering:

`http://editorial-host-ip-address:8080/escenic-admin/`

in the browser address field. The following page is displayed:



3. Click on **New publications**. The following page is displayed:



4. Click on the **Choose File** button and locate the **demo-temp-dev.war** file you have copied to your local machine.

5. Click on **Upload**. The following page is displayed:

Administration

- [Home](#)
- [List publications](#)
- [New publications](#)
- [Publication tools](#)

Uploading resources

This page allows you to upload resources to the server. Once you have uploaded the required resources, you will be able to create publications.

Create Publication

You now have enough resources to [create a publication](#)

Available resources

You have now uploaded the following resources. Once you have uploaded enough valid resources, you can create your publication.

Type	Age	Uploaded Content Type	File name	Size (bytes)	Valid	
/escenic/content-type	00	application/vnd.escenic.content-type	META-INF/escenic/publication-resources/escenic/content-type	6332	Validated	Remove uploaded resource
/escenic/feature	00	text/plain	META-INF/escenic/publication-resources/escenic/feature	380	Validated	Remove uploaded resource

6. Click on the **create a publication** link. The following page is displayed:

Administration

- [Home](#)
- [List publications](#)
- [New publications](#)
- [Publication tools](#)

The publication will be created based on the publication definition files you have uploaded. If you want to use different files, you can [upload different files](#).

Publication Name:

Note: The name must be a symbolic name, with no spaces or punctuation. The publication name is simply an administrative name for the site. For an intranet/internet server good examples include "intranet" or "internet"; while for a hosting site, consider using a symbolic customer name, e.g. "somecompany" or "johndoe"

Administrator password:

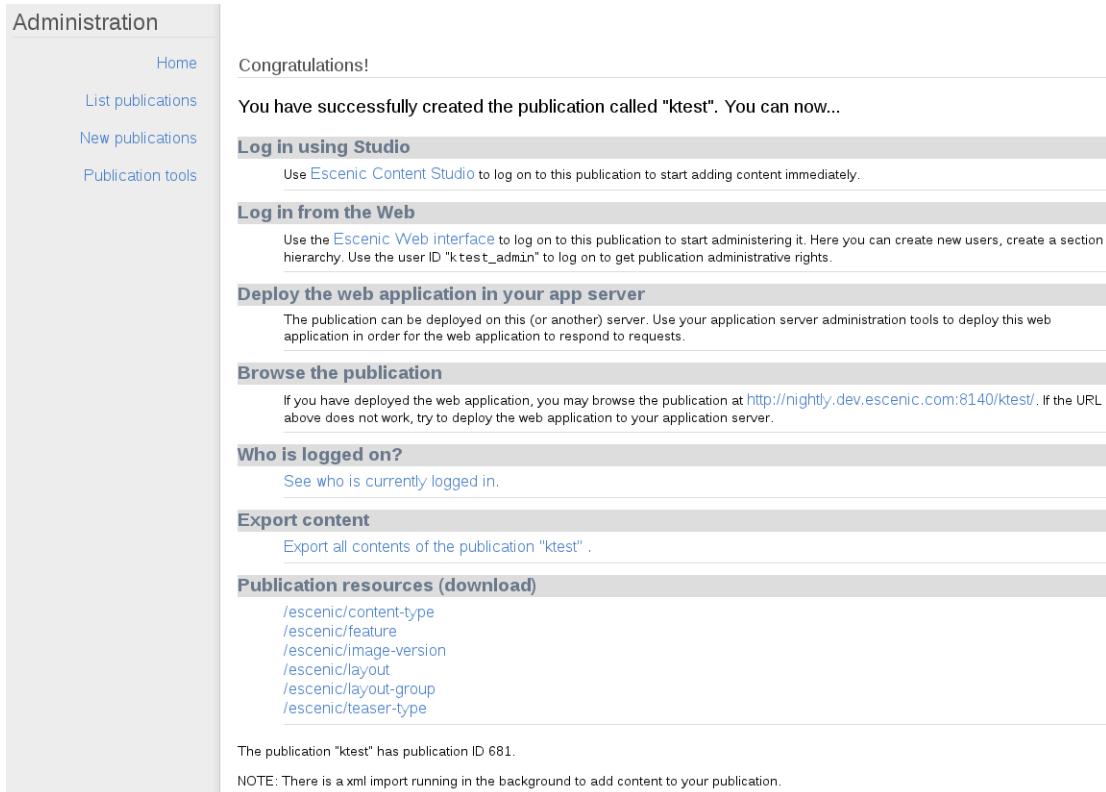
Verify password:

To access your publication after it has been created, an administrator user is created. To protect access to the publication, you can specify the administrator's password. Note: Take care of the administrator password.



Creating a new publication

A Publication is the Escenic representation of one complete web-site. Many installations only have one publication, while other installations have several publications. To decide whether or not your installation requires several publications, an easy rule of thumb is **one publication per web site**. Say Escenic is being used to power a company internet; you might only need one publication. If Escenic is used to power several company intranets, it is logical to create a Publication for each such web site. The same rule could apply to an installation with a Company Intranet and Extranet, where the two sites would primarily contain different data.

- Enter a name for the publication and an administrator password in the displayed form, then click on **Submit**. The following page is displayed:



The screenshot shows the 'Administration' page for a publication named 'ktest'. On the left is a sidebar with links: Home, List publications, New publications, and Publication tools. The main content area has a 'Congratulations!' message and instructions on how to log in using Studio, the Web interface, or by deploying the web application. It also provides a URL to browse the publication and a list of publication resources for download. At the bottom, it notes the publication ID is 681 and that an XML import is running in the background.

You have now created a publication. If you click on **Home** and then **Status** to redisplay the status pages, you should see that all the  warnings that were displayed earlier have now been replaced by  icons. This should be the case not only on the **editorial-host** where you are currently working, but on all your **engine-hosts**.

The **demo-temp-dev** publication is now present in the database, and can be edited, but cannot be viewed from a browser until the **demo-temp-dev** web application is deployed on the application server. To deploy **demo-temp-dev**, do the following on your **assembly-host**, while logged in as **escenic**:

- Copy **demo-temp-dev.war** to the assembly tool's **publications** folder:

```
$ cp /opt/escenic/engine/contrib/wars/demo-temp-dev.war /opt/escenic/assemblytool/publications/
```

- Create a text file called **demo-temp-dev.properties** in the **/opt/escenic/assemblytool/publications/** folder, with the following contents:

```
name: demo-temp-dev
source-war: demo-temp-dev.war
context-root: /demo-temp-dev
```

- Reassemble the Content Engine installation:

```
$ ece assemble
```

Then, on each **engine-host**, while logged in as **escenic**:

- If you are installing everything on one host, then skip this step.

On each **engine-host**, copy `/var/cache/escenic/engine.ear` from the **assembly-host**. If you have installed an SSH server on the **assembly-host** and SSH clients on your **engine-hosts**, then you can do this as follows:

```
$ scp -r escenic@assembly-host-ip-address:/var/cache/escenic/engine.ear /var/cache/escenic/
```

where *assembly-host-ip-address* is the host name or IP address of your **assembly-host**.

2. Redeploy the Content Engine:

```
$ ece deploy
```

3. Restart the Content Engine:

```
$ ece restart
```

4. Open a browser and enter the following URL in the address field:

```
http://engine-host-ip-address:8080/publication-name/
```

where *publication-name* is the name you gave to your **demo-temp-dev** publication. This should display the publication.

Publication web applications must be deployed on all hosts where the publications are to be viewed. This normally means all **engine-hosts** (including **editorial-hosts**, since writers and editors will want to preview their work). You may not want to deploy this test publication on all your hosts, however.

3.22 Test Web Studio

Now you have created a publication, you can use it to check that the Web Studio web application is correctly installed. To do this:

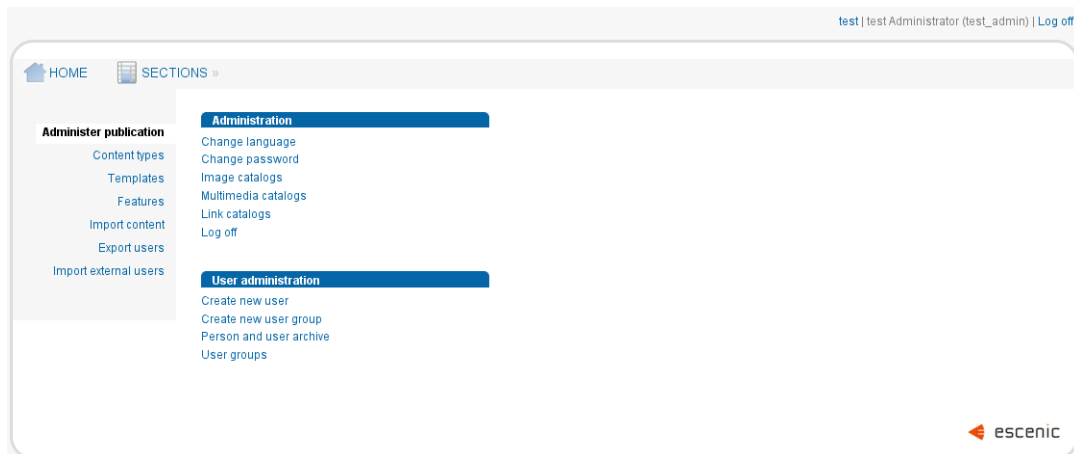
1. Open a browser, and access the Web Studio web application (called **escenic**) on one of your **editorial-hosts** by entering:

```
http://editorial-host-ip-address:8080/escenic/
```

in the browser address field. A login page should be displayed.

2. Enter the **demo-temp-dev** publication's administrator user name (*publication-name_admin*) and the administrator password you specified when creating the publication.

3. The following page should then be displayed:



3.23 Test Content Studio

To check that Content Studio is correctly installed:

1. Make sure that a Java runtime environment is installed on your computer. If necessary, download the latest version from <http://www.java.com> and install it.
2. Open a browser, and access the Content Studio download page on one of your **editorial-hosts** by entering:

| `http://editorial-host-ip-address:8080/studio/`

in the browser address field. A download page should be displayed.

3. Click on the **Launch Escenic Content Studio** button. After the application has downloaded, a login dialog should be displayed.
4. Enter the **demo-temp-dev** publication's administrator user name (*publication-name_admin*) and the administrator password you specified when creating the publication.
5. The Content Studio application should then be downloaded, and start up after a few moments. For information on how to use Content Studio, see the **Escenic Content Studio User Guide**.

4 Recommended Modifications

This section contains instructions for various additions and modifications you can make to the basic installation described in [chapter 3](#). These changes are recommended for production installations.

4.1 Configure SSL Support

A production installation will almost always need SSL support, in order to be able to provide secure access to the Content Engine for remote users of Content Studio, Web Studio and so on.

To set up SSL support you must:

1. Obtain a certificate from a Certificate Authority (CA) such as VeriSign or Thawte. (But see [section 4.1.1](#).)
2. Install the certificate on every **engine host** for which you require HTTPS access. This means you should at least install the certificate on all your **editorial hosts**. You might, however, also want to install it on your **presentation hosts** in order to provide HTTPS access to some or all of your published content. For detailed instructions on how to install certificates for use by Tomcat, see <http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html>.
3. Open `/opt/tomcat/conf/server.xml` for editing.
4. Somewhere in this file you will find a commented-out **Connector** element for configuring SSL connections on port 8443. Uncomment this element, and add **keystoreFile**, **keystorePass** and **URIEncoding** attributes to it as follows:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
  maxThreads="150" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS"
  keystoreFile="keystore-path" keystorePass="password" URIEncoding="UTF-8"/>
```

keystore-path must be the path of the keystore you created in step 2. *password* must be the keystore/certificate password you created in step 2. The **URIEncoding="UTF-8"** attribute is required on all **Connector** elements in order to ensure that Content Engine search functionality works for non-Latin characters.

5. Open the configuration layer file `com/escenic/webstart/StudioConfig.properties` for editing, and change the protocol name in the web service URL used by by Content Studio. That is, change the setting of `property.com.escenic.client.webservice.url` from something like this:

```
property.com.escenic.client.webservice.url=http:host-name:port/webservice/
index.xml
```

to something like this:

```
property.com.escenic.client.webservice.url=https:host-name:port/webservice/
index.xml
```

If you installed everything on one host, then you will need to do this in your common configuration layer (`/etc/escenic/engine/common/com/escenic/webstart/StudioConfig.properties`). If you have a multi-host installation, then you will need to do

it in one or more host configuration layers (`/etc/escenic/engine/host/host-name/com/escenic/webstart/StudioConfig.properties`).

The above procedure ensures that the Content Engine can support HTTPS access, but it does not enforce it in any way. Enforcement of HTTPS access to specific resources from specific locations can be achieved in a variety of ways and is outside the scope of this manual.

4.1.1 Using Self-Signed Certificates

The use of self-signed certificates is possible, but not recommended. Not only are self-signed certificates unverifiable, setting up Content Studio to work with self-signed certificates is complicated and error-prone.

A description of how to create and install self-signed certificates for Tomcat is included here: <http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html>. In order to use Content Studio in combination with self-signed certificates, however, it is also necessary to install certificates on all the computers on which Content Studio is run. For a description of how to do this, see http://blogs.sun.com/andreas/entry/no_more_unable_to_find.

You will also need to set two Java system properties on each machine where Content Studio is run:

javax.net.ssl.trustStore

The path of the keystore (or trust store) on the computer.

javax.net.ssl.trustStorePassword

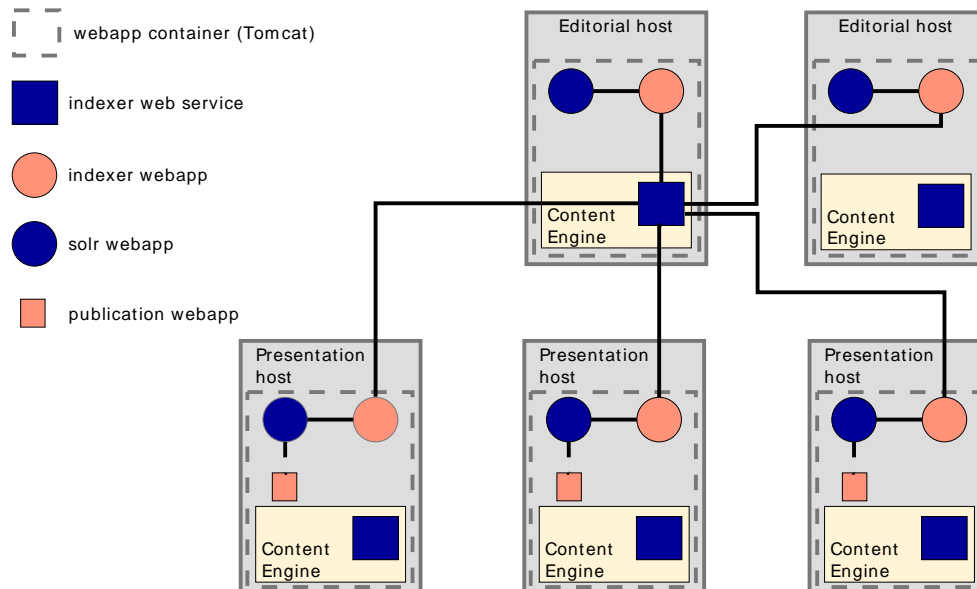
The keystore/certificate password you specified when adding the self-signed certificate. If you did not specify a password, the default is **changeit**.

If the certificates are identically installed on all computers from which Content Studio will be used, then you can include these settings in the `/com/escenic/webstart/StudioConfig.properties` file in your common configuration layer (see [section 3.15](#)). Otherwise they will need to be individually set on each computer.

4.2 Search Engine Deployment and Configuration

The Content Engine's search functionality is provided by the Java search engine Apache Solr, which runs as a web application. The standard Content Engine installation includes a **solr** web application and an associated **indexer** web application that indexes the content of all Escenic publications. These two applications are deployed along with the Content Engine by the **ece** script's **deploy** action.

The result of following the basic installation procedure described in [chapter 3](#), therefore, is that a **solr** instance and **indexer** web application is deployed on every **engine host** in your installation, all with identical configurations.



This set up will work, but it is relatively inefficient and is unlikely to work well in a production environment. There are two main reasons for this:

Solr memory usage

solr can at times consume large amounts of memory and trigger large garbage collection operations in the JVM, which has severe effects on Content Engine performance. It should therefore not be run in the same JVM as the Content Engine on production systems. The simplest way to achieve this separation on a single-host installation is to run **solr** and the **indexer** webapp in a separate Tomcat instance. For more about this, see **Escenic Content Engine Server Administration Guide, section 5.2.3**.

Solr stemming

In the default **solr** configuration, English stemming is enabled by default. This means that searching non-English content might give unexpected results.

If your content is in a language other than English, you should either disable stemming or modify the configuration to suit your language.

To disable stemming, remove the following line from **schema.xml**:

```
<filter class="solr.SnowballPorterFilterFactory" protected="protwords.txt"
  language="English"/>
```

Disabling stemming will improve **solr**'s performance.

For information about how to configure stemming for other languages, see the Solr documentation on <http://lucene.apache.org/solr/>.

Solr optimization issues

The default **solr** configuration is optimized for editorial purposes: it indexes all the fields needed to support the search functionality provided by Content Studio, resulting in very large indexes. This is acceptable in the editorial context, since the number of concurrent Content Studio users, even in a very large organisation, is not likely to be very large. The **presentation**

hosts in a large Escenic installation, however, can be required to serve many thousands of concurrent users, and the default **solr** configuration may perform poorly in this context.

The default configuration, therefore, is fine for the **editorial hosts** in a production system, but for the **presentation hosts** you are recommended create a custom indexer configuration that only indexes the fields actually needed to support the kinds of search required in your publications.

To do this, open `var/lib/escenic/schema.xml` for editing on each of your **presentation hosts**, and modify the index schema to meet your requirements. Editing this file is outside the scope of this manual. In order to tune the search engine you need to take account of the both the contents of your publications, your users' needs with regards to search and the limitations imposed by your particular hardware configuration. For further information and advice on tuning, see the Solr documentation on <http://lucene.apache.org/solr/>.

There are many more changes you can make to your search engine set-up in order to optimize it for your particular needs. For a discussion of the general principles involved, see **Escenic Content Engine Server Administration Guide, chapter 5**.

4.3 Distributed Memory Cache

A multiple-host installation in which each Content Engine instance maintains its own cache is not particularly efficient: it is better to pool the memory in a single, common cache. The most commonly-used means of doing this is to use the open source distributed cache manager **memcached**.

The instructions below tell you how to install **memcached**, configure it and then reconfigure your publications' **PresentationArticleCache** caches to use it. These caches will then share a single memory pool managed by **memcached**.

These instruction are based on the assumption that you want to install **memcached** on all your **engine-hosts**: **memcached** will then use some memory from each host. It might be the case, however, that you would prefer to concentrate the cache on a smaller number of hosts, or on one specific host. You may even wish to locate the cache on a completely different host, one that is not running the Content Engine. **memcached** is a completely free-standing application, and you can install it on any host or group of hosts in your network. The configuration described below is only one of many possible set-ups.

Note that if you are using **memcached**, the decorator must be serializable.

4.3.1 Install memcached

1. Install **memcached** on all your **engine hosts**.

On Debian-based Linux machines, you can do this by logging in as **root** and entering the following command:

```
# apt-get install memcached
```

2. Open `/etc/memcached.conf` for editing. Find the following line:

```
-l 127.0.0.1
```

and comment it out:

```
# -l 127.0.0.1
```

This allows **memcached** to listen on all interfaces.

3. **memcached** is installed as a daemon and will be started automatically on system reboot. However, to start it now without rebooting, enter:

```
# /etc/init.d/memcached start
```

4.3.2 Install the memcached Java Libraries

In order to make **memcached** accessible to Java web applications, you need to also install a Java client library on all your engine hosts. To do this:

1. Log in on your **assembly-host** as **escenic**.
2. Download the Java client library for **memcached** from <http://www.whalin.com/memcached/#download> to a temporary location.

```
$ cd /tmp/
$ wget http://img.whalin.com/memcached/jdk5/log4j/java_memcached-
release_2.0.1.tar.gz
```

3. Unpack the downloaded package:

```
$ tar xzf java_memcached-release_2.0.1.tar.gz
```

4. Copy the unpacked **memcached** JAR file to the **/opt/escenic/assemblytool/lib**:

```
$ cp java_memcached-release_2.0.1/java_memcached-release_2.0.1.jar \
> /opt/escenic/assemblytool/lib/
```

5. Assemble and redeploy the Content Engine as described in [section 3.18](#).

4.3.3 Configure memcached

The following instructions tell you to make changes in your common configuration layer, because they are based on the assumption that you want all your Content Engines to share a common cache. If this is not the case, then the following actions should be carried out in the host configuration layers of the hosts you want to use **memcached**.

To configure **memcached** you need to add a **.properties** file to your common configuration layer.

1. Log in as **escenic** on one of your **engine hosts**.
2. Create a folder for the new properties file:

```
$ mkdir -p /etc/escenic/engine/common/com/danga/
```

3. Create a file called **SocketIOPool.properties** in the new folder, open it for editing and add the following content:

```
$class=com.danga.MemCached.SocketIOPool
# fill in memcached servers here.
servers=host-name1:11211,host-name2:11211

# how many connections to use
initConn = 10
minConn = 5
maxConn = 100
# idle time
maxIdle = 180000
maintSleep = 5000
# socket timeout
```

```
socketTO = 3000
failover = true
# a network lookup algorithm
nagle = false
```

where *host-name1*, *host-name2*, etc. are replaced by the names of the hosts on which you have installed **memcached**.

The above parameter settings should work satisfactorily in most cases. Should you need to modify them, consult the documentation available on the memcached web site (<http://memcached.org/>).

4. Open **/etc/escenic/engine/common/Initial.properties** for editing and add the following lines:

```
# using memcached
service.0.0-memcached-socket-pool=/com/danga/SocketIOPool
```

4.3.4 Configure PresentationArticleCache Instances

The final step is to configure **PresentationArticleCache** in all publication web applications to use **memcached**. To do this you need to:

1. Add a **/WEB-INF/localconfig/neo/xredsys/presentation/cache/** folder to each publication web application. For example:

```
$ mkdir -p webapp/WEB-INF/localconfig/neo/xredsys/presentation/cache/
```

This will need to be done on the system(s) where your publication web applications are developed and maintained. For general information about publication development and maintenance, see the **Escenic Content Engine Template Developer Guide**.

2. Create a file called **PresentationArticleCache.properties** in each of the new folders, and add the following content to each file:

```
$class=neo.util.cache.Memcached
```

3. Redeploy the publication web applications in the usual way (see the **Escenic Content Engine Template Developer Guide** for general information about this).

It is also possible to configure this service so that hitting **flushCache** also flushes the **memcached** cache. This is not recommended!

```
flushMemCached=true
```

With this set, hitting **flushCache** on any of the servers connected to the same **memcached** cluster will flush the complete **memcached** cache!