Change Log Daemon

Developer Guide

2.0.0-3

escenic

A CCI COMPANY

Table of Contents

1 Introduction

The Change Log Daemon for Escenic Content Engine provides a means of tracking changes made in the Escenic Content Engine database. It can be used in the implementation of a wide range of extensions to Escenic applications, such as:

- Exporting content from the Content Engine to some other system (a print system for example).
- Importing content from an external system when certain types of change occur.
- Processing content as it is added to the Content Engine (advanced text analysis for example).

In all these cases, the Change Log Daemon ensures that the extension is provided with a complete record of all changes made to content, so that it can take appropriate action.

There are other ways of watching what happens in the Content Engine - you can, for example, use event listeners or transaction filters. The Change Log Daemon, however, has a number of advantages:

- You can write your action or agent (chapter 4) in any language.
- An agent is completely decoupled from the Escenic Content Engine, and runs in a separate JVM. If you create more than one agent then each one runs in its own JVM. This makes it easier for your agent to have a different release cycle than the rest of your cluster. You can also restart a malfunctioning agent without affecting the rest of the cluster.
- It is scalable. One agent puts no more load on the Escenic Content Engine than a Content Studio user, in some cases less.
- It follows the principle of doing only one task and doing it well: one Change Log Daemon instance may have only one agent.
- You don't need to write your own parser and logic to read the change log or follow links or change log pages. All of this is done for you by the Change Log Daemon.
- Resilience is built in. The Change Log Daemon keeps track of the changes your agent has handled and which agent operations failed, permanently or temporarily, and will always pick up where it left off after a restart.

1.1 Conventions

The instructions in the following section assume that you have a standard Content Engine installation, as described in the **Escenic Content Engine Installation Guide**.

The Content Engine and the software it depends on may be installed on one or several host machines depending on the type of installation required. In order to unambiguously identify the machines on which various installation actions must be carried out, the **Escenic Content Engine Installation Guide** defines a set of special host names that are used throughout the manual.

Some of these names are also used here:

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.

In addition we define one additional host type:

changelog-daemon-host
The machine(s) used to host Change Log Daemon(s).

The host names always appear in a bold typeface.

2 Installation

This chapter contains step-by-step instructions for installing Change Log Daemon on a single host computer.

2.1 Preconditions

The following preconditions must be met before you can install the Change Log Daemon:

- The Content Engine has been installed as described in the **Escenic Content Engine Installation Guide** and is in working order.
- You have the required distribution file **`changelog-daemon-2.0.0-3.zip`**.
- The JDK installed on your **changelog-daemon-host**(s).

2.2 Download

Download the Change Log Daemon distribution from the Escenic documentation web site (http://documentation.vizrt.com). If you have a multi-host installation with shared folders as described in the **Escenic Content Engine Installation Guide**, then it is a good idea to download the distribution to your shared **`/mnt/download`** folder:

```
$ cd /mnt/download
$ wget https://user:password@maven.escenic.com/com/escenic/changelog/
changelog/2.0.0-3/changelog-2.0.0-3.zip
```

Otherwise, download it to some temporary location of your choice.

2.3 Unpack Change Log Daemon components

On your **`changelog-daemon-host`**, while logged in as root:

1. Create a folder for Change Log Daemon under /opt/escenic:

   ```
   $ mkdir /opt/escenic/
   $ chown -R escenic:escenic /opt/escenic/
   ```

2. Change user to **escenic** and unpack Change Log Daemon package as follows:

   ```
   $ su - escenic
   $ cd /opt/escenic/
   $ unzip /mnt/download/changelog-2.0.0-3.zip
   ```

The new folder contains several different files and directories

changelog.sh
: A shell script for starting the changelog

changelog.jar
: The Jar file that contains everything you need to run the daemon

documentation
Related documentation

examples
Examples on how to configure a Change Log Daemon

A Change Log Daemon can only drive one agent, so if you want to do several different things based on the change log, you will need to run multiple instances of the Change Log Daemon.

To install more than one Change Log Daemon on the same **changelog-daemon-host**, unpack the distribution in several directories and repeat the installation process for all daemons.

2.4 Create default configuration layer

The default configuration layer will be created automatically the first time the daemon is started.

On the **changelog-daemon-host**, while logged in as **escenic**:

```
$ cd /opt/escenic/changelog-daemon-2.0.0-3
$./changelog.sh
 -
  - ********************************
  - Sample configuration files were created in
  -    /opt/escenic/changelog-daemon-2.0.0-3/config
  - Please see the documentation for more information on how to configure this service
  - ********************************
 -
```

2.5 Configure the daemon

On **changelog-daemon-host**, while logged in as **escenic**, open **`/opt/escenic/changelog-daemon-2.0.0-3/config/Daemon.properties`** for editing and set the following properties:

- **`url`**
- **`username`**
- **`password`**

For instructions on how to set these properties (and other properties in the **`Daemon.properties`** file), see section 6.1.

2.6 Configure an agent

The Change Log Daemon is now almost ready to go - it just needs an agent to do some actual work. You can either use one of the ready-made agents supplied with the Change Log Daemon or create your own. For information on how to do this, see Usage of Agent (chapter 4).

2.7 Start the daemon

On the **changelog-daemon-host**, while logged in as **escenic**:

```
$ cd /opt/escenic/changelog-daemon-2.0.0-3
$./changelog.sh
```

3 Directory layout

This chapter gives an overview of all the folders and files in a running Change Log Daemon system.

The list below lists directories and files created by a default setup. Any configuration changes might change the location of files.

Additional files and folders might be created by the agents. These will not be listed below.

`classes`
Java class files that will be added to the beginning of the classpath when the Change Log Daemon is started

`config`
The configuration layer. This directory will normally contain two files, **Daemon.properites** and **Agent.properites**. See chapter 6 for an overview of all the available configuration options.

If these files are not present, the Change Log Daemon will not start.

`documentation`
Documenting on how to install and configure Change Log Daemon

`examples`
Example configuration

`lib`
Jar files that will be included in the classpath when the Change Log Daemon is started

`log`
The log files

`permanent-errors`
Changelog entries that cannot be handled by the configured agent.

`temporary-errors`
Changelog entries that could not be processed. The daemon will try to process them at a later stage.

`.state.properties`
Used by the Change Log Daemon to keep track of it's current state. If this file is deleted, the Change Log Daemon will start processing the changelog using the provided entry point.

4 Creating and Using Agents

Five different implementations of the **`Agent`** interface are provided with the Change Log Daemon. There are two ready-to-use classes plus three abstract classes on which you can base your own implementations. You can also create your own agent implementation from scratch, but this is not recommended.

4.1 Using a Ready-Made Agent

To use one of the ready-made agents, all you need to do is configure it correctly and add a reference to one of your Change Log Daemon instances.

4.1.1 com.escenic.changelog.agent.FileAgent

This agent generates XML files containing change log entries. One file is created for every entry added to the change log. The files are written to a folder specified in the agent configuration file. The files are given auto-generated names consisting of a timestamp plus the extension **`.xml`**.

4.1.1.1 Configuration and usage

The easiest way of configuring a **FileAgent** is to use the example configuration included in the distribution.

On the **changelog-daemon-host**, while logged in as **escenic**:

```
$ cd /opt/escenic/changelog-daemon-2.0.0-3
$ cp examples/file/config/Agent.properties config/
```

This configuration will work out of the box.

For a full overview of all the configuration options for this Agent, see section 6.2

4.1.2 com.escenic.changelog.agent.ExecuteAgent

This agent allows you to write your agent code in any language you choose. Every time an entry is added to the change log, it calls a program specified in the agent configuration file. The program is run in an external process and the agent writes the content of the change log entry to the process's standard input as an XML stream. Return values from the external process are handled as follows:

`0`
: Success

`1`
: The agent throws a **`PermanentException`**.

Any other value
: The agent throws a **`TemporaryException`**.

4.1.2.1 Configuration and usage

The easiest way of configuring an **ExecuteAgent** is to use the example configuration included in the distribution.

On the **changelog-daemon-host**, while logged in as **escenic**:

```
$ cd /opt/escenic/changelog-daemon-2.0.0-3
$ cp examples/shell/config/Agent.properties config/
```

The default configuration will try to invoke the Linux command **cat** for each changelog entry. The output of this command will be written to the log. It will use the current directory as the working directory for the command.

To change the command, open the **`config/Agent.properties`** for editing and change

```
command=cat
```

to

```
command=<path_to_a_script>
```

where **<path_to_a_script>** is the path to the script you would like to invoke for each changelog entry.

For a full overview of all the configuration options for this Agent, see section 6.3

4.2 Creating an Agent

Three abstract classes that you can use as the basis for your own agent are included in the Change Log Daemon distribution. The only difference between them is the form in which change log entries are supplied. Extending all three involves implementing the same four methods.

The available extension points are

com.escenic.changelog.AbstractAgent
: This is the base implementation. See AbstractAgent (section 7.1) for details

com.escenic.changelog.agent.XOMAgent
: This implementation supplies change log entries as XOM element objects. See XOMAgent (section 7.2) for details

com.escenic.changelog.agent.JdomAgent
: This implementation supplies change log entries as JDOM element objects. See JDomAgent (section 7.3) for details

We recommend extending one of these classes when implementing your own agent.

4.2.1 Extending AbstractAgent

This chapter will give you the basics needed to create your own agent. It will show you how to create a sublass of **`com.escenic.changelog.AbstractAgent`**, how to compile it and how to configure it.

The agent in this example will just print everything provided by the daemon to **System.out**, but even though it's pretty simple it will give you the building boxes needed to create agent's with more functionality.

4.2.1.1 Creating the Java class

On the **changelog-daemon-host**, while logged in as **escenic**:

```
$ cd /opt/escenic/changelog-daemon-2.0.0-3
$ mkdir src
$ touch src/Agent.java
```

Open **Agent.java** in a text editor insert the following

```
public class SystemOutAgent extends com.escenic.changelog.AbstractAgent {
  @Override
  protected void consumeEntry(final java.io.InputStream pInputStream) throws
 com.escenic.changelog.PermanentException, com.escenic.changelog.TemporaryException {
    try {
      org.apache.commons.io.IOUtils.copy(pInputStream, System.out);
    }
    catch (java.io.IOException e) {
      throw new com.escenic.changelog.TemporaryException("Could not write to
 System.out. Will retry later");
    }
  }

  @Override
  protected void startService() throws IllegalStateException,
 IllegalArgumentException, Exception {
  }

  @Override
  protected void stopService() throws IllegalStateException, Exception {
  }
}
```

4.2.1.2 Compile agent

The next step is to compile the agent.

On the **changelog-daemon-host**, while logged in as **escenic**:

```
$ javac -classpath changelog.jar:lib/*.jar -d classes src/SystemOutAgent.java
```

> If you need any additional Jar files while compiling your source code, these Jar files must be added to the **`lib`** folder. If not, the Jar file will not be available runtime and the daemon will not start.

4.2.1.3 Configure the agent

On the **changelog-daemon-host**, while logged in as **escenic**, open **`config/Agent.properties`** and add

```
$ $class=SystemOutAgent
```

5 Logging

The logging framework used by Change Log Daemon is **`log4j`**. This chapter explains where the log files are located and how to increase the log levels.

5.1 Log file

The log for Change Log Daemon can be found in **changelog.home/log**.

5.2 Increase log level

The default log level is **`ERROR`**

The log level can be changed by adding

```
$log=<log level>
```

to the property file of the component you would like to debug.

If you for instance want to change the logging level to **`debug`** for the **Agent**, open **`config/Agent.properties`** ,add

```
$log=DEBUG
```

save the file and restart the Change Log Daemon.

Valid log levels are

- ERROR
- WARNING
- DEBUG
- INFO
- TRACE

6 Configuration reference

6.1 Daemon

All Change Log Daemon configuration settings are stored in one file called **`Daemon.properties`**.

The **`Daemon.properties`** can be found in the **`/opt/escenic/changelog-daemon-2.0.0-3/config/`** folder. It contains the following property settings:

`url`
Set this to point to the Content Engine change log you want to use. For example:

```
url=http://editorial-host-ip-address/webservice/escenic/changelog/
publication/publicationId
```

Where *editorial-host-ip-address* is the host name or IP address of your **editorial-host**, and *publicationId* is the id of the publication you are interested in.

`username`
A Content Engine user name. This user name will be used log into the Content Engine web service and access the change log.

`password`
The password for the specified user.

`agent`
The nursery path to the agent we should use when consuming the change log. This defaults to **`/com/escenic/daemon/Agent`**, so you only need to have a **`classes/com/escenic/daemon/`** directory and put your **`Agent.properties`** there.

`temporaryErrorsFolder`
The folder in which temporary failures are stored. The default is **`.temporary-errors`**.

`permanentErrorsFolder`
The folder in which permanent failures are stored. The default is **`.permanent-errors`**.

`direction`
The direction in which log entries are to be read. Allowed values are:

`previous` (default)
From oldest to newest

`next`
From newest to oldest

`pollInterval`
The number of seconds between attempts to check the change log. The value must be larger than **`0`**. The default is **`10`**.

`bootstrapDelay`
The number of seconds to wait after start-up before checking the change log for the first time. The value must be larger than **`0`**. The default is **`5`**.

`temporaryErrorPollInterval`
The number of seconds between attempts to check the temporary errors folder (in order to retry). The value must be larger than `0`. The default is `60`.

6.2 FileAgent

`directoryPath`
Defines the directory where the files are stored. The agent supports both absolute and relative paths.

If the directory does not exists, the agent will try to create it. If creating the directory fails, the agent will not start.

6.3 ExecuteAgent

`command`
The full path to a file that will be executed once per changelog entry. The file must be executable.

If the path is not absolute, the file must be available in path.

`deletedEntryCommand`
The full path to a file that will be executed once per deleted changelog entry. The file must be executable.

If the path is not absolute, the file must be available in path.

If this property is not defined, the agent will use the script defined in the **command** property.

`directory`
The working directory where the script will be executed.

7 Class reference

7.1 com.escenic.changelog.AbstractAgent

This is the base implementation. It supplies change log entries as unprocessed Atom XML entries. To extend it, implement the following four methods:

`startService()`
This method is called when the agent is started. Use it to validate the agent configuration, log into third-party services (if required) and carry out any other necessary start-up operations.

The following exceptions are handled by the framework:

`IllegalStateException`
Throw this exception if the agent is in the wrong state for start-up.

`IllegalArgumentException`
Throw this exception if the agent configuration is invalid.

`Exception`
Throw this exception if anything else goes wrong during startup.

`consumeEntry(final InputStream pInputStream)`
This method is called once for each entry in the change log. It is called by **`handleEntry(final InputStream pInputStream)`**, but only if **`isServiceRunning()`** returns **`true`**. The **`InputStream`** contains an XML representation of this content item.

The following exceptions are handled by the framework:

`PermanentException`
Throw this exception if an error occurs that will not be corrected without external intervention (from the system administrator, for example). The Change Log Daemon will not resubmit this entry for processing.

`TemporaryException`
Throw this exception if an error occurs that is likely to be temporary. The Change Log Daemon will resubmit this entry later.

Any other exception thrown will be logged, and the Change Log Daemon stopped.

`consumeDeletedEntry(final InputStream pInputStream)`
This method is called once for each deleted entry in the change log. It is called by **`handleEntry(final InputStream pInputStream)`**, but only if **`isServiceRunning()`** returns **`true`**. The **`InputStream`** contains an Atom tombstone deleted-entry element.

The following exceptions are handled by the framework:

`PermanentException`
Throw this exception if an error occurs that will not be corrected without external intervention (from the system administrator, for example). The Change Log Daemon will not resubmit this entry for processing.

`TemporaryException`
Throw this exception if an error occurs that is likely to be temporary. The Change Log Daemon will resubmit this entry later.

Any other exception thrown will be logged, and the Change Log Daemon stopped.

`stopService()`
This method is called when the agent is stopped. Use it to log out of third-party services (if necessary) and carry out any other necessary clean-up operations.

The following exceptions are handled by the framework:

`IllegalStateException`
Throw this exception if the agent is in the wrong state for shut-down.

`IllegalArgumentException`
Throw this exception if the agent configuration is invalid.

`Exception`
Throw this exception if anything else goes wrong during startup.

7.2 com.escenic.changelog.agent.XOMAgent

This implementation supplies change log entries as XOM element objects, so you don't have to parse the XML yourself. To extend it, implement the following four methods:

`startService()`
This method is called when the agent is started. Use it to validate the agent configuration, log into third-party services (if required) and carry out any other necessary start-up operations.

The following exceptions are handled by the framework:

`IllegalStateException`
Throw this exception if the agent is in the wrong state for start-up.

`IllegalArgumentException`
Throw this exception if the agent configuration is invalid.

`Exception`
Throw this exception if anything else goes wrong during startup.

`consumeEntry(final Element pEntry)`
This method is called once for each entry in the change log. It is called by **`handleEntry(final InputStream pInputStream)`**, but only if **`isServiceRunning()`** returns **`true`**. **`pEntry`** contains an XOM **`Element`** representing the change log entry.

The following exceptions are handled by the framework:

`PermanentException`
Throw this exception if an error occurs that will not be corrected without external intervention (from the system administrator, for example). The Change Log Daemon will not resubmit this entry for processing.

`TemporaryException`
Throw this exception if an error occurs that is likely to be temporary. The Change Log Daemon will resubmit this entry later.

Any other exception thrown will be logged, and the Change Log Daemon stopped.

`consumeDeletedEntry(final Element pEntry)`
This method is called once for each deleted entry in the change log. It is called by **`handleEntry(final InputStream pInputStream)`**, but only if **`isServiceRunning()`**

returns **true**. **pEntry** contains an XOM **Element** representing an Atom tombstone deleted-entry element.

The following exceptions are handled by the framework:

PermanentException
Throw this exception if an error occurs that will not be corrected without external intervention (from the system administrator, for example). The Change Log Daemon will not resubmit this entry for processing.

TemporaryException
Throw this exception if an error occurs that is likely to be temporary. The Change Log Daemon will resubmit this entry later.

Any other exception thrown will be logged, and the Change Log Daemon stopped.

stopService()
This method is called when the agent is stopped. Use it to log out of third-party services (if necessary) and carry out any other necessary clean-up operations.

The following exceptions are handled by the framework:

IllegalStateException
Throw this exception if the agent is in the wrong state for shut-down.

IllegalArgumentException
Throw this exception if the agent configuration is invalid.

Exception
Throw this exception if anything else goes wrong during startup.

7.3 com.escenic.changelog.agent.JdomAgent

This implementation supplies change log entries as JDOM element objects, so you don't have to parse the XML yourself. To extend it, implement the following four methods:

startService()
This method is called when the agent is started. Use it to validate the agent configuration, log into third-party services (if required) and carry out any other necessary start-up operations.

The following exceptions are handled by the framework:

IllegalStateException
Throw this exception if the agent is in the wrong state for start-up.

IllegalArgumentException
Throw this exception if the agent configuration is invalid.

Exception
Throw this exception if anything else goes wrong during startup.

consumeEntry(final Element pEntry)
This method is called once for each entry in the change log. It is called by **handleEntry(final InputStream pInputStream)**, but only if **isServiceRunning()** returns **true**. **pEntry** contains a JDOM **Element** representing the change log entry.

The following exceptions are handled by the framework:

`PermanentException`
Throw this exception if an error occurs that will not be corrected without external intervention (from the system administrator, for example). The Change Log Daemon will not resubmit this entry for processing.

`TemporaryException`
Throw this exception if an error occurs that is likely to be temporary. The Change Log Daemon will resubmit this entry later.

Any other exception thrown will be logged, and the Change Log Daemon stopped.

`consumeDeletedEntry(final Element pEntry)`
This method is called once for each deleted entry in the change log. It is called by **`handleEntry(final InputStream pInputStream)`**, but only if **`isServiceRunning()`** returns **`true`**. **`pEntry`** contains a JDOM **`Element`** representing an Atom tombstone deleted-entry element.

The following exceptions are handled by the framework:

`PermanentException`
Throw this exception if an error occurs that will not be corrected without external intervention (from the system administrator, for example). The Change Log Daemon will not resubmit this entry for processing.

`TemporaryException`
Throw this exception if an error occurs that is likely to be temporary. The Change Log Daemon will resubmit this entry later.

Any other exception thrown will be logged, and the Change Log Daemon stopped.

`stopService()`
This method is called when the agent is stopped. Use it to log out of third-party services (if necessary) and carry out any other necessary clean-up operations.

The following exceptions are handled by the framework:

`IllegalStateException`
Throw this exception if the agent is in the wrong state for shut-down.

`IllegalArgumentException`
Throw this exception if the agent configuration is invalid.

`Exception`
Throw this exception if anything else goes wrong during startup.