Section Feed

Plug-in Guide

3.1.1-1

Table of Contents

1 Introduction

The Section Feed plug-in automatically cross-publishes content items. It executes a set of **`feeds`**. A **`feed`** selects content items that meet specified criteria from a **`source`** section, and publishes them in a **`destination`** section.

All the information needed to carry out these operations is stored in a special Content Store publication resource an XML publication resource called a **`section-feed`** resource. Like most of the standard Content Store publication resources described in the Escenic Content Engine Resource Reference, the menu resource is an XML file. Once you have created a **`section-feed`** resource for a publication you can upload it to the Content Store in the same way as you upload other resources, using the **`escenic-admin`** web application (for details, see http://docs.escenic.com/ece-server-admin-guide//upload_resources1.html).

When the Section Feed plug-in is installed and started, it will listens for relevant events generated by the Content Store: events such as a content item being created, a content item being added to a new section and so on. When a significant event occurs, the Section Feed plug-in checks the **`section-feed`** resource to see if the event should trigger a cross-publishing action, and if so carries out the action.

1.1 Using the Section Feed Plug-in

Once you have installed and configured the Section Feed plug-in as described in chapter 2, all you need to do start using it is:

1. Create a **`section-feed`** resource file defining the cross-publishing rules you want the plug-in to follow. For a detailed description of the **`section-feed`** resource format, see chapter 3.
2. Upload the resource using **`escenic-admin`** as described in http://docs.escenic.com/ece-server-admin-guide//upload_resources1.html.

2 Installation

The following preconditions must be met before you can install Section Feed 3.1.1-1:

- The Content Store and CUE assembly tool have been installed as described in the CUE Content Store Installation Guide and are in working order.
- You have the required distribution file **`section-feed3.1.1-1.zip`**.

2.1 Conventions

The instructions in the following section assume that you have a standard Content Store installation, as described in the CUE Content Store Installation Guide. *escenic-home* is used to refer to the **`/opt/escenic`** folder under which both the Content Store itself and all plug-ins are installed.

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 CUE Content Store Installation Guide defines a set of special host names that are used throughout the manual.

Some of these names are also used here:

assembly-host
The machine used to assemble the various Content Store components into an enterprise archive or .EAR file.

engine-host
The machine(s) used to host application servers and Content Store instances.

editorial-host
engine-host(s) that are used solely for (internal) editorial purposes.

The host names always appear in a bold typeface. If you are installing everything on one host you can, of course, ignore them: you can just do 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.

2.2 Section Feed Installation

Installing Section Feed involves the following steps:

1. Log in as **`escenic`** on your **assembly-host**.
2. Download the Section Feed distribution. If you have a multi-host installation with shared folders as described in the CUE Content Store 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/plugins/section-feed/
   section-feed/3.1.1-1/section-feed-3.1.1-1.zip
   ```

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

3. If the folder **`/opt/escenic/engine/plugins`** does not already exist, create it:

   ```
   $ mkdir /opt/escenic/engine/plugins
   ```

4. Unpack the Section Feed distribution file:

   ```
   $ cd /opt/escenic/engine/plugins
   $ unzip /mnt/download/section-feed-3.1.1-1.zip
   ```

   This will result in the creation of an **`/opt/escenic/engine/plugins/section-feed`** folder.

5. Log in as **`escenic`** on your **assembly-host**.

6. Run the **`ece`** script to re-assemble your Content Store applications.

   ```
   $ ece assemble
   ```

   This generates an EAR file (**`/var/cache/escenic/engine.ear`**) that you can deploy on all your **engine-host**s.

7. If you have a single-host installation, then skip this step.

   On each **engine-host** on which you wish to run Section Feed, 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-host`**s, 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**. Note that although the Section Feed plug-in may be installed on multiple hosts, it should only be **enabled** on one host at any one time (see section 2.4).

8. Deploy the EAR file and restart the Content Store on each **engine-host** by entering:

   ```
   $ ece stop
   $ ece deploy
   $ ece start
   ```

2.3 Verify The Installation

To verify the status of Section Feed, open the CUE Admin web application (usually located at **`http://`***`server`***`/admin`**) and click on **View installed plugins**. The status of all currently installed plug-ins is shown here, and indicated as follows:

The plug-in is correctly installed.

The plug-in is not correctly installed.

2.4 Configuration

After installing Section Feed (see chapter 2) you need to configure it to meet your requirements. This involves copying a single configuration file from the Section Feed installation into the host configuration layer for at least one of your Content Store hosts and then modifying the copied file to meet your requirements. In detail, you must:

1. Log in to your Content Store host as the **`escenic`** user.
2. Copy the supplied common configuration layer files as follows:

   ```
   $ cp -r /opt/escenic/engine/plugins/section-feed/misc/siteconfig/com/escenic \
   > /etc/escenic/engine/host/host-name/com
   ```
3. Open **`/etc/escenic/engine/host/`***host-name***`/com/escenic/section-feed/SectionFeed.properties`** for editing and set the following properties:

   `serviceEnabled`

   Set this to **`true`** on one (and only one) of your hosts. If you have installed Section Feed on multiple hosts then it must be set to **`false`** on all other hosts. If Section Feed is enabled on multiple hosts at the same time, then your feed instructions will be executed multiple times.

   `feedHiddenArticles`

   Set this property to **`false`** if you want Section Feed to ignore hidden content items.

3 section-feed

This chapter describes the XML format used to store section feed definitions.

Namespace URI

The namespace URI of the **section-feed** schema is .

Root Element

The root of a **section-feed** file must be a **feeds** element.

3.1 destination

Represents a publication section that is to be used as a feed destination. Selected content items are automatically cross-published to this section. Exactly where the cross-published content items are added is determined by the child **placement** element.

Syntax

```
<destination
    sectionid="..."
  >
  <placement/>
</destination>
```

Attributes

sectionid="..."
The id of the destination section.

3.2 destination-group

A group of **destination** elements. You can create a **destination-group** element and refer to it using the **ref-destination-group** element.

Syntax

```
<destination-group
    name="..."
  >
  <destination>...</destination>*
</destination-group>
```

Attributes

name="..."
The name of the **destination-group**.

3.3 feed

The **feed** element defines a single feed action. It contains child elements that together specify the content items that are to be cross-published from a source section (or group of source sections) to a destination section (or group of destination sections).

Syntax

```
<feed
    name="..."
  >
  (<ref-source-group/>|<ref-destination-group/>|<source>...</
source>|<destination>...</destination>)*
</feed>
```

Attributes

name="..."
The name of the **feed**.

3.4 feeds

The root element of a **section-feed** resource.

Syntax

```
<feeds>
  (<source-group>...</source-group>|<destination-group>...</destination-
group>|<feed>...</feed>)*
</feeds>
```

3.5 filter

Represents a filter used when selecting content items from a **source** section. Only content items that satisfy the filter's requirements will be selected for cross-publishing. If a **source** element has several child **filter** elements, then a content item will be selected if it satisfies any one of them.

Syntax

```
<filter
    type="(page|section|list|inbox)"?
  />
```

Attributes

type="(page|section|list|inbox)" (optional)
The type of the **filter**.

Allowed values are:

page (default)
Only content items that are added to the active page of the section will be selected.

`section`
: Only content items that are added to the section itself will be selected.

`list`
: Only content items that are added to one of the section's lists will be selected.

`inbox`
: Only content items that are added to one of the section's inboxes will be selected.

3.6 placement

Specifies where in a **`destination`** section cross-published content items are added.

Syntax

```
<placement
    poolid="..."
    type="(inbox|list)"?
    placement="(top|bottom)"?
  />
```

Attributes

`poolid="..."`
: The id of the inbox or list to which content items should be added.

`type="(inbox|list)"` (optional)
: The pool type to which content items should be added.

  Allowed values are:

  `inbox` (default)
  : Add cross-published content items to an inbox.

  `list`
  : Add cross-published content items to a list.

`placement="(top|bottom)"` (optional)
: Specifies where content items are to be added.

  Allowed values are:

  `top` (default)
  : Add content items to the top of the specified list/inbox.

  `bottom`
  : Add content items to the bottom of the specified list/inbox.

3.7 ref-destination-group

A reference to a **`destination-group`** element.

Syntax

```
<ref-destination-group
    name="..."
    />
```

Attributes

`name="..."`
The name of the **`ref-destination-group`**.

3.8 ref-source-group

A reference to a **`source-group`** element.

Syntax

```
<ref-source-group
    name="..."
  />
```

Attributes

`name="..."`
The name of the **`ref-source-group`**.

3.9 source

Represents a publication section that is to be used as a feed source. Content items added to this section will be automatically cross-published to another section. Exactly how content items are selected for cross-publishing can be limited by adding child **`filter`** elements.

Syntax

```
<source
    sectionid="..."
  >
  <filter/>*
</source>
```

Attributes

`sectionid="..."`
The id of the source section.

3.10 source-group

A group of **`source`** elements. You can create a **`source-group`** element and refer to it using the **`ref-source-group`** element.

Syntax

```
<source-group
    name="..."
```

```
  >
  <source>...</source>*
</source-group>
```

Attributes

`name="..."`
: The name of the **`source-group`**.