Section Feed

Plug-in Guide

2.3.1.153084

escenic

A CCI COMPANY

Table of Contents

1 Introduction

The **`Section Feed`** plug-in will automatically feed articles into sections, or in other words cross publish the article. Whenever an article meets certain criteria for a feed, it will be placed wherever that feed says.

A **feed** defines which articles go where. It contains two components. A **source**-section and a **destination**-section. A feed will be performed when an article is added to the source section. This plug-in will add (or cross publish) the article to the destination section.

All information about which section to get articles from and to which section the articles be fed to is stored as XML as a publication resource called **`/escenic/plugin/section-feed`**. A visual interface allows editing of the the automatic feed information so you don't need a knowledge of XML. All of this will be explained in detail later on.

1.1 Structure

This document has the following structure

- We start documentation with an overview of the SectionFeed component. This is general explanation of how the component works.
- Then there is a chapter on how to install it. This chapter must be read together with the Chapter 5.
- Next there is a user guide. This will describe how to create a new feed.
- At the end there is a general plugin installation guide.

1.2 Audience for this guide

This documentation contains everything from installation guide to user guide. We think all users should read Chapter 1 to get a general overview.

- **`User`** - a user need to know how to edit the Section Feed by using the graphical user interface. For information about this see Chapter 3.
- **`Installer`** - see Chapter 2 and Chapter 5.
- **`System administrator and advanced users`** - should probably(or want to) have a brief knowledge about every thing.

1.3 Technical requirements

The following requirements must be met if you want to install and run the **`SectionFeed`** plug-in.

Required version of Escenic Content Engine: 5.0-6 or higher. For previous versions of Escenic use an earlier version of the section feed plug-in.

Required plug-ins:

- xml-editor plug-in, version 2.0-1 or higher

1.4 What is Section Feed?

The **`Section Feed`** plug-in is a component that automatically feed articles from one section into another section. The article then is cross published.

To create a feed you need two sections. One that can serve as a source section and one that can be the destination section. A feed will be performed when an article is added to the source section. Then the article will simultaneously be added(or cross published) to the destination section.

1.5 What purpose does SectionFeed serve?

SectionFeed is a component used to automatically feed articles into sections. This might save the user some work. For instance if you have a section with several sub sections, and you want the parent section to always have the latest articles from all sub sections you could use a feed.

Lets say you have section structure like this: the section **`Sport`** with the sub sections **`Golf`**, **`Tennis`** and **`Football`**. Then you could create a feed that will automatically cross publish articles to **`Sport`** that are published into either one of the sub sections.

1.6 How does SectionFeed work?

When the SectionFeed plug-in is installed and started, it will listen to events sent by the Escenic Content Engine. These kind of event will for instance be created when a new article is created, when an article is added to a new section, when an article get a new relation and so on. This plug-in will only listen to events where articles are added to new sections or new desks.

When an event of this type is sent the SectionFeed look up in a XML-file to see if this a event to act on. If it is, the plug-in will feed the this article to the destination section defined in the XML-file.

Don't worry. You do not need any XML knowledge. This plug-in has an user-friendly interface to create and manage feeds. This interface will be explained in Chapter 3.

There is one special case: what if the article already is in the specified destination section? Then the article will not be added.

2 Installation

The first step is an ordinary plug-in installation. See Chapter 5 for detailed description.

The standard installation will add the service **`/com/escenic/sectionfeed/SectionFeed`** to the **`/Initial`** component of the local configuration layer.

2.1 Additional configuration

The SectionFeed plug-in requires some additional installation steps to complete the installation process. These steps are described below.

2.1.1 Enable the SectionFeed component

It is necessary to enable the service. The plug-in installs a component called **`/com/escenic/sectionfeed/SectionFeed`** and this component must be enabled for it to operate correctly. Create the file **`/etc/escenic/engine/common/com/escenic/sectionfeed/SectionFeed.properties`** add the line

serviceEnabled = true

This must only be done on one server in a cluster of servers.

> In a cluster of Escenic Content Engine servers, the SectionFeed component must be enabled on only one of the servers otherwise your articles might be fed several times and you get duplicates.

2.2 Additional configuration options

It is possible to specify that the section feed should ignore so-called **hidden** articles. This is done by setting the **`feedHiddenArticles`** property to false. See the example .properties file on how to do this.

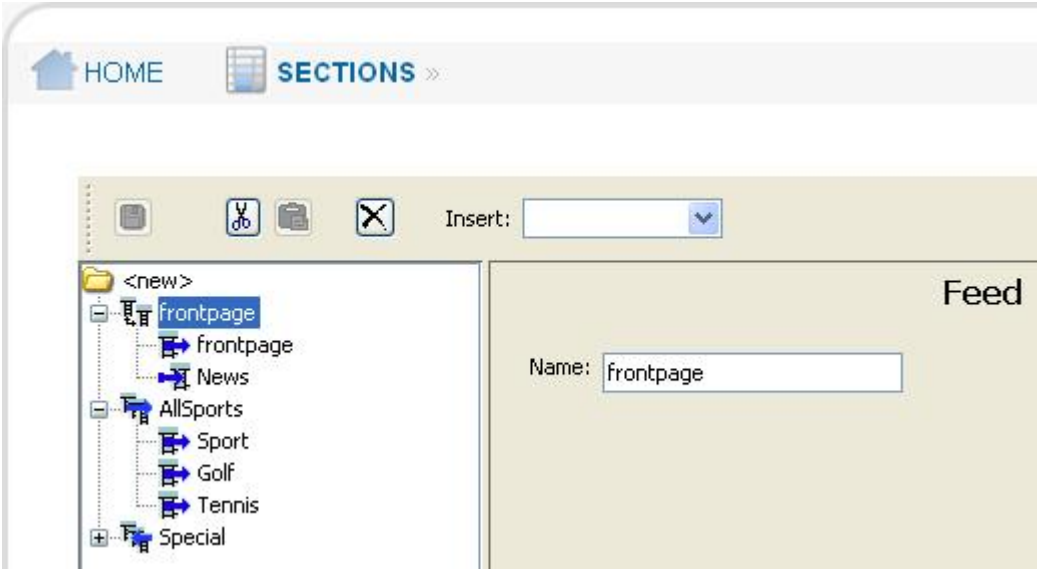
3 Using the section feed editor

This chapter describes how to edit the section feed definitions with the Web Studio section feed editor applet.

The editor is available from the main page of the Escenic Web Interface (**`components - section feed`**). To use the editor, the browser must have a java-plugin installed and enabled.

3.1 Overview

The editor has three main elements. The toolbar, structure area and an edit area. See figure below.



On top there is a toolbar with the most common operations such as save, cut, copy, paste and insert.

The left part of the editor display the structure of the feeds. Here you can create new feed, source and destination elements, or edit existing.

The right half is the edit area. Here you can edit all attributes of the currently selected element in the the structure area. The edit area will change depending on the selected item. The figure above display a **`feed`** element.

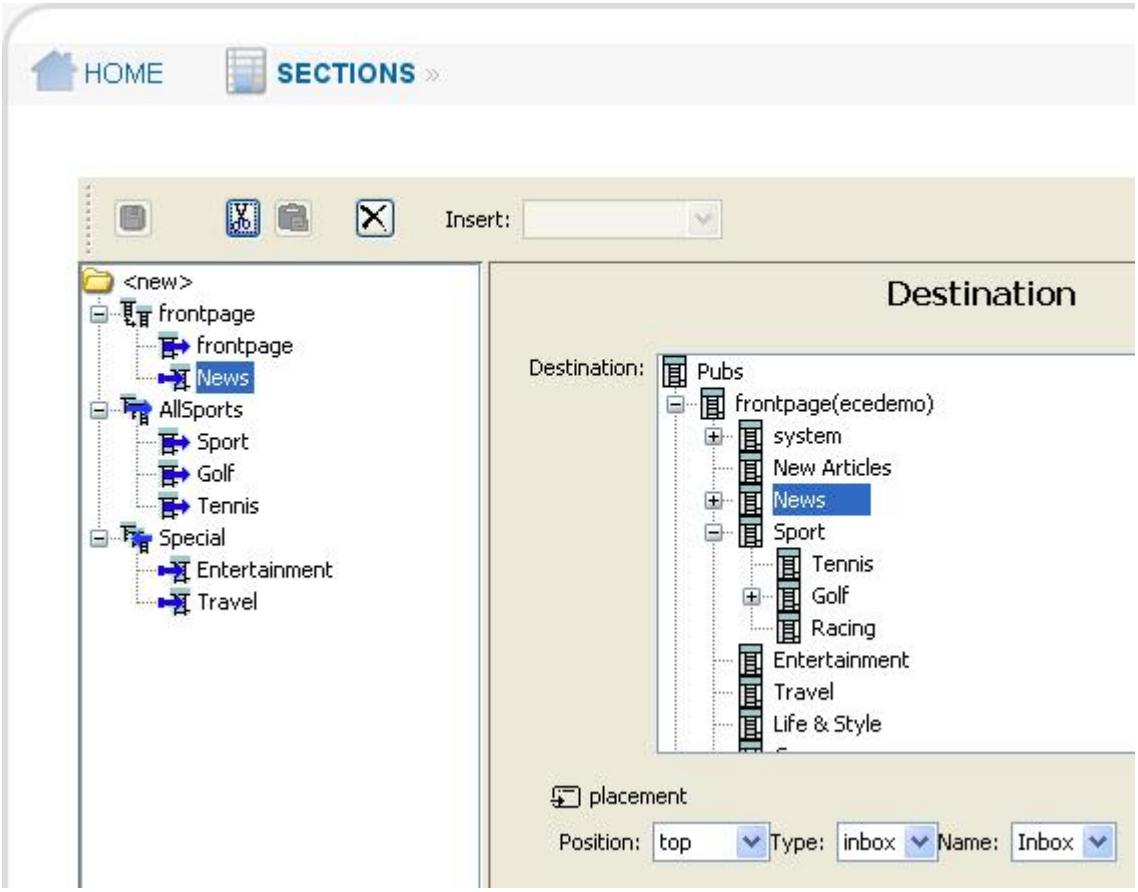
3.2 Using the section feed editor

All editing of feeds can be done either by using the toolbar or the right-click mouse menu. The two menus give the same options. Available operations are:

- **Inserting an element**: Select the element where to insert a new element. Choose which element to insert (feed, source, destination, source-group or destination-group) from the dropdown menu.
- **Cut an element**: It will be placed on the 'clipboard'.

- **Paste an element**: Only available if there is an element on 'clipboard'
- **Deleting an element**: Select the element to delete. Click 'x' on toolbar.

To edit an element just left click on the element, and all attributes will be visible in the edit area to the right.



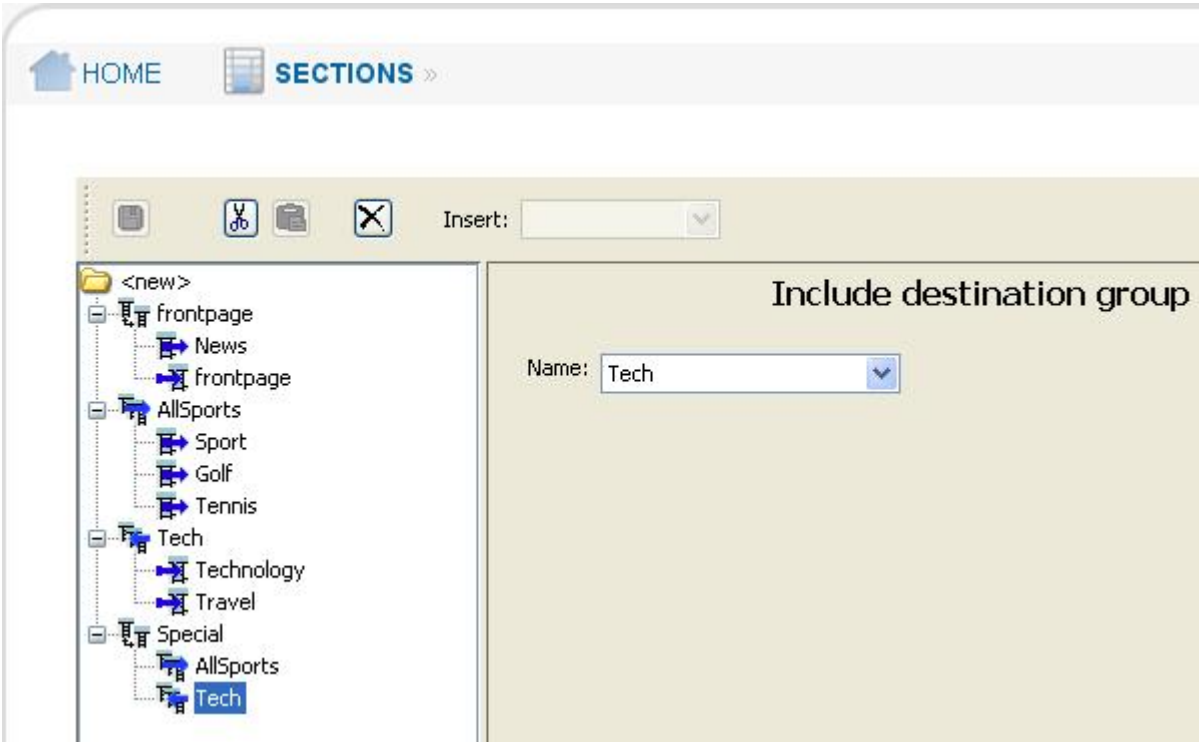
Icons

Which icons do we have, and do they mean? The usage of al elements will be explained later on in this documentation.

- - feed
- - destination
- - destination group
- - source
- - source group
- - filter
- - placement

Structure

How is the structure? I will try to explain this by using two simple examples. To have a valid feed element, is must contain both a source and destination element.



In the figure above I have created two feeds. In the first there is just three elements: **`feed`**, **`source`** and **`destination`**. As source I have used the section News, and as destination the frontpage.

So when somebody publish an article to the **News** section, this article will automatically be added to the **frontpage**.

The second feed is a bit more complex. The **`feed`** name special contains a **`source-group`** and a **`destination-group`**. The groups include must of course also be defined. In my case this is done just above the feed element. I have called my source group AllSports and the destination-group Tech.

Here the same thing will happen as in the simple feed. When an article is added to the **Tennis** section, it will automatically also be added to the **Technology** section.

3.3 Elements

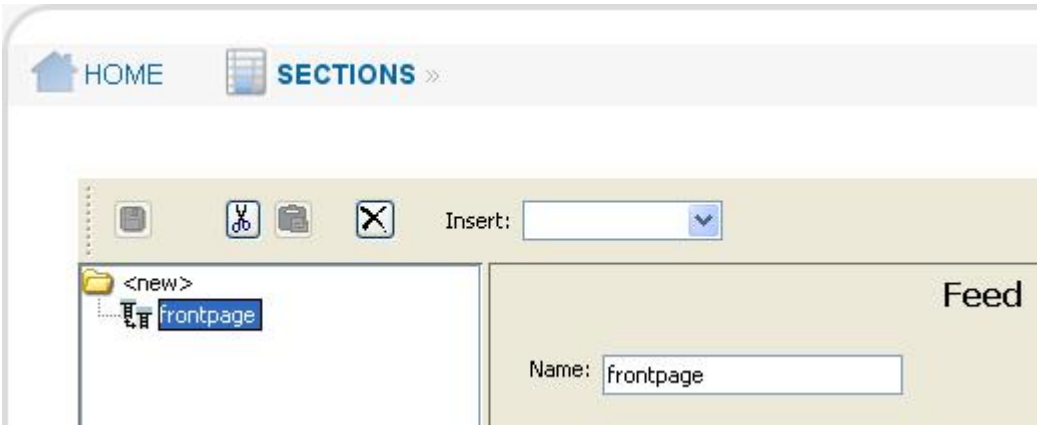
This will be a short explanation of all elements used.

3.3.1 Feed

The **`feed`** is the key element here. It will contain both **`source`** and **`destination`**. When you add a new **`feed`**, the only attribute you have to set is the name of the feed. Then you can add the wanted elements. In its simplest form a **`feed`** should contain at least one **`source`** and one **`destination`**.

Valid elements in a feed:

- **source**
- **destination**
- **ref-source-group**: name of the source-group to include in this feed
- **ref-destination-group**: name of the destination-group to include in this feed



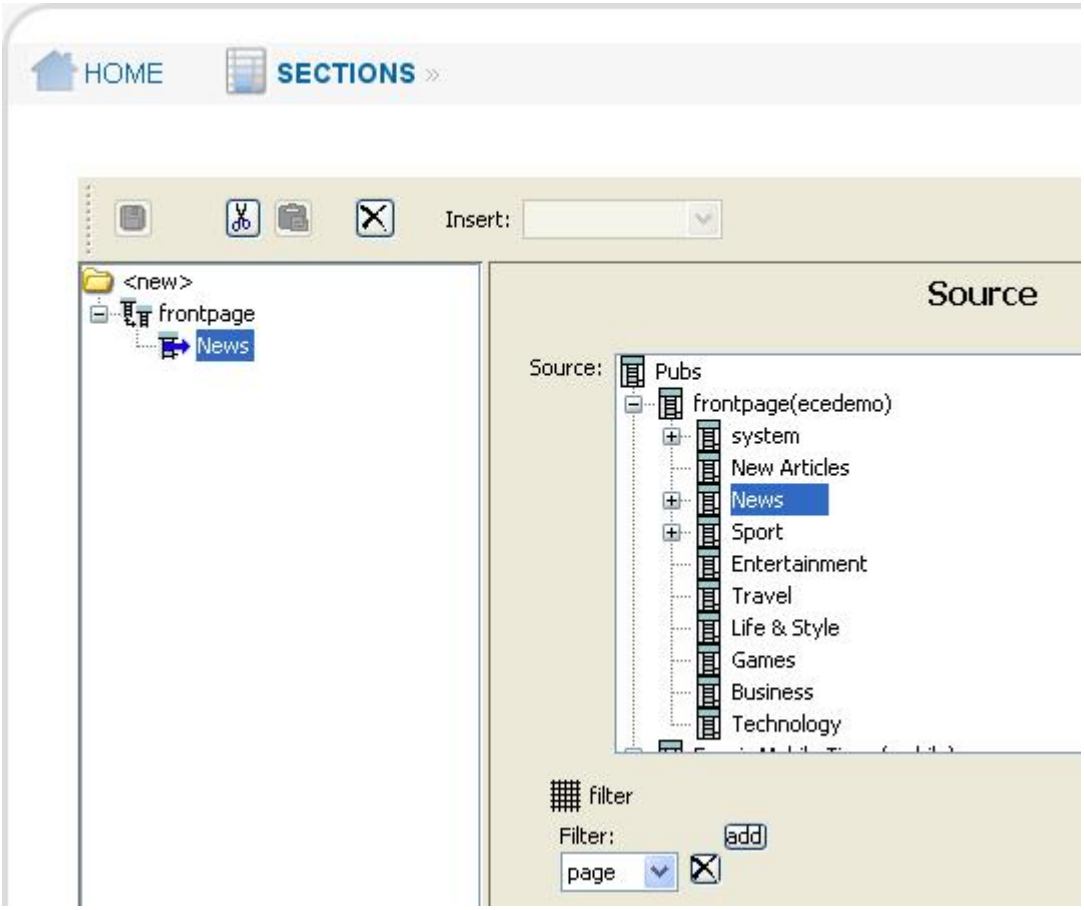
3.3.2 Source

When you add a **source** to the feed, a section tree will appear to the right. Click on the wanted section. Below you can choose a filter to be used on the source.

The filter option will filter away all other options the selected. If you select **page**, the section will only be used as source when an article is added to the active **page**. Possible values:

- **page**: articles added to the active index page
- **section**: articles added to the section
- **list**: articles added to any of the lists

- **inbox**: articles added to any of the inboxes



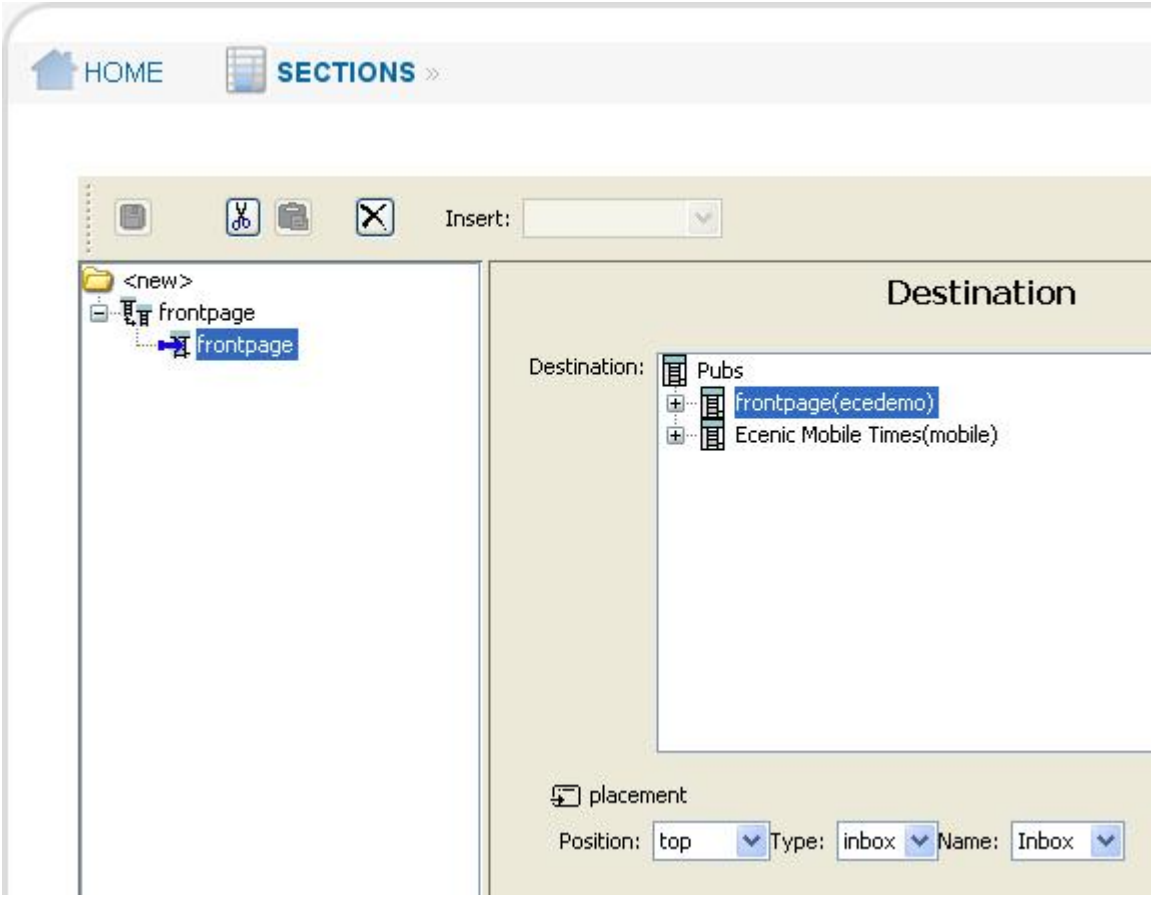
3.3.3 Destination

The same thing will happen when you add a **`destination`** as when adding a **`source`**. The section tree will appear in to the right. Select the section to be used as destination for the feed.

Now you have to select the **`placement`**. It will determine where the article will be placed. Here there are three select boxes:

- **type**: To which type of pool. There are two options: inbox and list.
- **name**: To which inbox or list the articles will be added.

- **placement**: Which place in the specified inbox or list the articles will be added. There are two options: top and bottom.



3.3.4 Source Group

There are two reasons to use a source group.

First it make it easier to reuse sources in a **`feed`**. Lets say you have a several sources you want to include in several feeds. Then you can create a **`source-group`** and include the desired sources. Next you can include the created **`source-group`** in all the feeds you want.

Second you can use it to simplify the **`feed`** element. If you have many sources in a feed, you can create a source group as a container to hold the group of source elements. Then include it into the feed.

3.3.5 Destination Group

The same as source group, there is two reasons to use a destination group. Either reuse of the destination elements, or to simplify the feed.

3.4 Create a feed

We will now create a feed step by step.

To create a working feed there are 3 elements that are mandatory:

- feed
- source
- destination

3.4.1 Simple feed

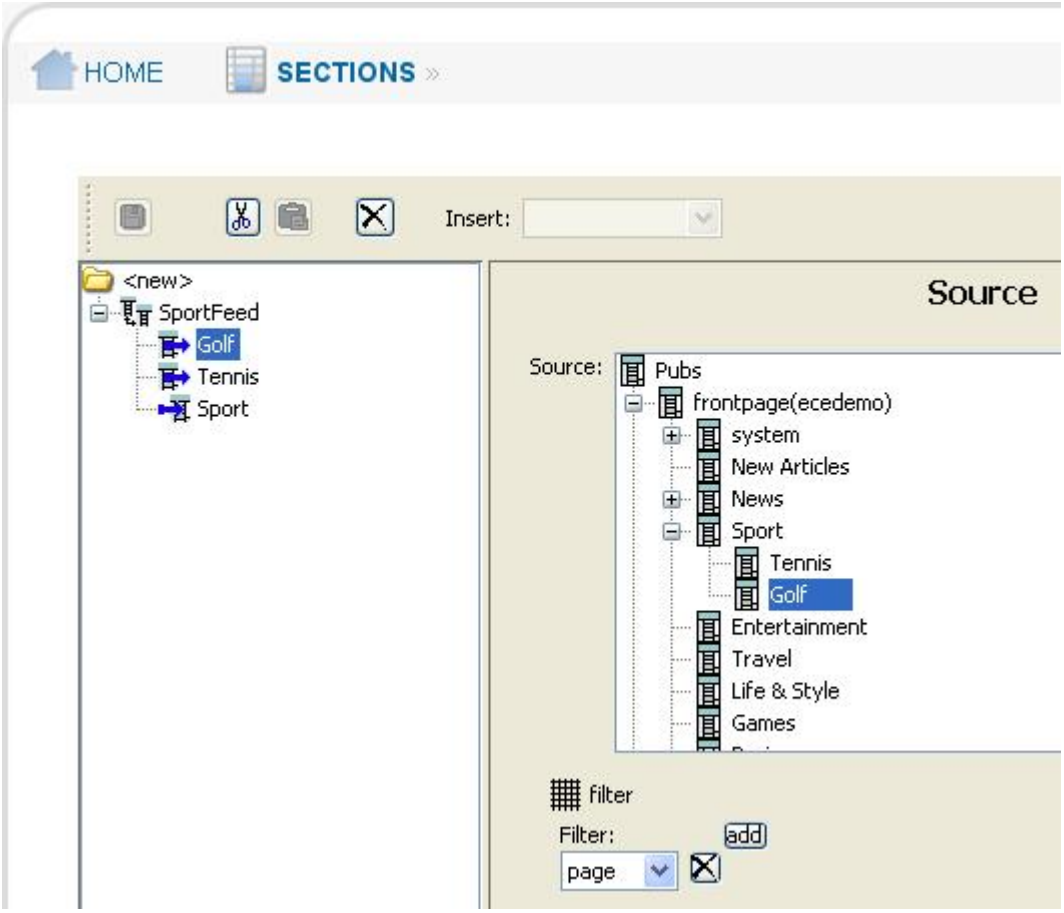
First element we need is a **`feed`** element. As you now know you have two possible ways to insert elements. In this guide I will use the right-click option.

Right-click on the root, and insert a feed element. Now the feed symbol is displayed with the name ***<new>**. In the edit area the name field is empty. Then I give it a name. In my case: **SportFeed**

Next I will create a destination. Right-click on the **`feed`**, and insert a destination. Then the edit windows will open a section browser. There I select the **Sport** section. This feed I want to put articles on the **`inbox`** named **`Inbox`** and position **`top`**. All these selections will I do in the editor window. Use the three dropdown menus: **Type**, **Name** and **Position**.

Now we have defined the feed and the destination. The last element we need is one or more sources. In this case it will be two: **Tennis** and **Golf**

Right click on the **feed** element, and choose insert source. Then the section tree will be displayed in the edit area. Select the wanted section. Here you also can set a filter. In this example it will be set to **page**. Repeat these steps to add more sources(as I will to add both **Tennis** and **Golf**).



This example will look as the figure above. Here you can see the **feed** element with the name SportFeed, the two **source** element(Golf and Tennis) and the **destination** Sports.

The feed we now has created will cross publish all articles that are added to the active index page of either **Tennis** or **Golf** to the **Sports** section. The articles that are fed will be added at the **top** of the inbox named **Inbox**.

4 section-feed

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

The following sections explains the section-feed XML tags and attributes. For the formal definition of the section-feed XML format, please refer to the section-feed DTD located in **plugins/sectionFeed/escenic/webapp/plugin/sectionFeed** in your escenic installation.

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.

4.1 destination

Specifies the **destination** section of the **feed**.

Syntax

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

Attributes

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

4.2 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**.

4.3 feed

The **feed** element.

Syntax

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

Attributes

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

4.4 feeds

The root element.

Syntax

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

4.5 filter

The **filter** of **source** section.

Syntax

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

Attributes

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

Allowed values are:

page (default)
Filters if the article is added to the active page of the section.

section
Filters if the article is added to the section.

list
Filters if the article is added to any of the lists of the section.

inbox
Filters if the article is added to any of the inboxes of the section.

4.6 placement

Specifies where to feed the article in the **destination** section.

Syntax

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

Attributes

poolid="..."
The id of the pool (inbox or list) where to feed the article.

type="(inbox|list)" (optional)
The type of the pool.

Allowed values are:

inbox (default)
Feed the article to a inbox.

list
Feed the article to a list.

placement="(top|bottom)" (optional)
Specifies where to add the article in the pool.

Allowed values are:

top (default)
Adds the article at the top of the pool.

bottom
Adds the article at the bottom of the pool.

4.7 ref-destination-group

Refers to a **destination-group** element.

Syntax

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

Attributes

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

4.8 ref-source-group

Refers to a **source-group** element.

Syntax

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

Attributes

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

4.9 source

Specifies the **source** section of the **feed**.

Syntax

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

Attributes

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

4.10 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**.

5 Plug-in installation

This chapter is an installation guide for Escenic plug-ins.

- Escenic is installed and in working order.
- The EAR assembly tool has been extracted and successfully used to set up a test ear file as per the instructions in the ECE Installation Guide.
- The plug-in distribution file: **`<plugin>-<version>.jar`** (or **`.zip`**).

5.1 Installation

Escenic Plug-ins should be installed using the assembly tool available from the ECE download page on Technet.

5.1.1 Server Install

Create Plug-in Directory

If **`ECE_HOME/plugins`** do not exist, create it. If you want to change the location of the plug-in folder, edit the **`<assembly-tool-home>/assemble.properties`** file.

Unpack Plug-in Distribution

Unpack the plug-in distribution file to **`ECE_HOME/plugins`**, such that there is a directory **`plugins/<plugin-name>`** afterwards.

Run DB Scripts

Run database scripts from **`plugins/<plugin-name>/misc/database/<db-vendor>`**. If there is no **`misc/database`** directory there are no db scripts. The scripts must be run in this order:

1. **`tables.sql`**
2. **`constraints.sql`**
3. **`constants.sql`**
4. **`indexes.sql`**
5. **`history.sql`**

The last script, **`history.sql`**, is not critical and can be skipped if it fails.

Rebuild and Deploy ECE

Build the ECE Enterprise Archive using the command **`ant ear`**. The assembly tool will add the plug-in to ECE's classpath, including the default plug-in configuration. It will add any required components to the web applications. Deploy the new EAR-file including updated webapps.

Customize the Plug-in Configuration

Refer to Chapter 2 for any configuration options available to this plug-in.

> Some plug-ins contain a **`<plugin-name>/misc/siteconfig`** folder. Prior to ECE 4.3 the content of this folder had to be copied to the local configuration layer and changes to Initial.properties applied. This is no longer necessary. The plug-ins default configuration is

read from the classpath. The **siteconfig** folder, if it exists, contains skeleton configuration files that may be used to customize the plug-in.

Verify Installation
Restart the application server and verify the installation status in **View installed plugins** in the Escenic Admin webapp.

If the application server does not support EAR based deployment, the JAR files located in the **<plugin-name>/lib** folder must be added to the application server's classpath. All the WAR files that have been rebuilt with the assembly tool should be redeployed.

5.1.2 Installing Plug-in Web Applications

Some Escenic plug-ins come with their own administrator web applications. If available, these are located in the **<plugin-name>/webapps/** directory of the plug-in distribution. These will be included in the EAR by the assembly tool and will be available after deployment. The web applications may also be deployed manually by using the war file located in **ECE_HOME/dist/war/**. Please refer to your application server documentation on how to deploy a web application.

Excluding the Plug-in Web Wpplication From the EAR

The plug-in web application can be excluded from the EAR by using the property **global-exclude-webapps** in **assemble.properties** or by using the property **exclude-webapps** in the server class configuration.

The Statistics Plug-in

The web applications for the statistics plug-in are not included in the EAR by default. They can be included by removing them from the property **global-exclude-webapps** in **assemble.properties**.

5.1.3 Installing Demo Templates

Some Escenic plug-ins come with example templates. If available, these are located in the **<plugin-name>/wars/** directory of the plug-in distribution. To install the demo templates, do the following:

Create Publication
Upload the WAR file to the Escenic Admin webapp as a publication definition. Create a publication. If any content is bundled with the templates, an import will be started automatically.

Build Publication
Add a publication definition to your **/publications** folder, defining the publication name and the WAR file location. An example definition for the Calendar demo templates may look like this:

```
source-war:   ../../plugins/calendar/wars/calendardemo.war
context-root: /calendar
```

Save the file and build using the **ant -q ear** command. Note that the relative path in the above example depends on your setting for the **publication-wars** property of the assembly tool configuration.

Deploy the Publication
Deploy the complete EAR-file, which includes the new publication, or deploy the single WAR file located in the **`ECE_HOME/dist/war`** directory.

Development Setup

In a development environment, the publications should be deployed as exploded webapps. After building the demo templates the first time, extract the resulting WAR file located in the **`<assemblytool>/dist/war`** directory to an appropriate location and deploy this as an exploded webapp.

Redeploying a Webapp

OC4J, Tomcat, Resin
Overwrite the existing 'war' file with the new version and restart the appserver.

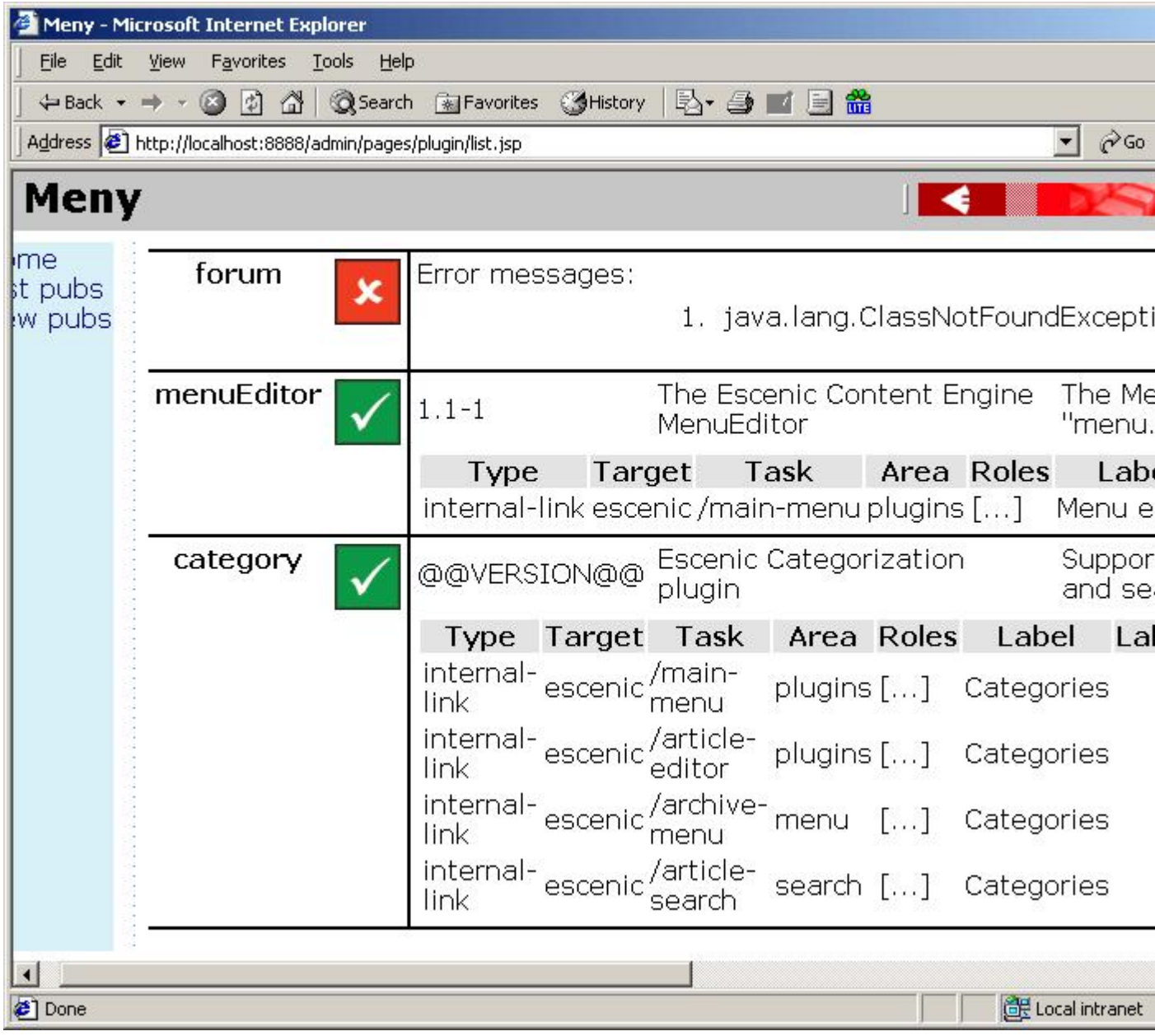
WebLogic
Use the 'console' web application (default: **`http://<server>:7001/console`**):

1. undeploy the running webapp (uncheck the 'deploy' box)
2. deploy the new 'war' file

5.2 Verifying the installation

Installation status can be checked in the Escenic **admin** webapp (default: **`http://<server>/admin`**). The link **`View installed plugins`** lists all installed plug-ins and their status. 'Green check' means OK, 'red cross' means that there is an installation problem.

The 'installation status' page does not check for all possible problems. When the installation status shows green, the installation should be checked further by logging in to Escenic, accessing the plug-in, and creating and deleting a few elements.



The 'category' and 'menuEditor' plug-ins are successfully installed. The 'forum' plug-in shows a 'ClassNotFoundException', meaning that there is a problem with the appserver classpath.

5.3 Troubleshooting

This section lists the most common problems and describes how to fix them.

ClassNotFoundException
On the plug-in status page or in the appserver log: The plug-in is not correctly installed or not installed at all. Check the contents of the **`engine.ear`** file and verify that all the libraries located in **`<plugin-name>/lib`** are present in the EAR file's lib folder.

404 Not Found
In the browser, when accessing a plug-in-related link: This may be caused by the webapp not being updated with the plug-in files. Please check that the corresponding WAR file has been updated by the assembly tool and redeploy the webapp if necessary. Note that the correct WAR file to deploy is located in the assembly tool's **`dist/war`** folder, not in ECE_HOME/webapps.

ClassCastException, NoSuchMethodError
In the appserver log or in the browser: A code version issue - there is a mismatch between the code version of Escenic and the code version of the plug-in. Contact Escenic for an update.

For other problems contact Escenic support.