Newsroom

Plug-in Guide

trunk-SNAPSHOT

Table of Contents

1 Introduction

A broadcast newsroom system is used to create, edit and manage news rundowns and control playout. Newsroom is a plug-in for Escenic Content Engine that provides an interface to the Avid **iNEWS** and **Enps** newsroom system. The Content Engine's Newsroom plug-in allows news items created in a newsroom system to be automatically imported into the Content Engine. As stories are saved in a newsroom, they automatically appear as content items in associated Escenic publications.

Assuming that the newsroom system is fully integrated with a Escenic Viz Content Pilot system then text, video and graphics components of a newsroom story will be transferred to the Content Engine and appear in the related publications.

The Newsroom plug-in is highly configurable. You can predefine:

- Which stories are to be transferred to the Content Engine.
- Which Escenic content types are to be used for newsroom stories.
- Which publication(s) newsroom stories are to appear in.
- Which section of a publication newsroom stories are to be added to.
- Which status (draft, published, etc.) newsroom stories are to be assigned when they are transferred to the Content Engine.
- Which inbox and which lists newsroom stories are to be added to.

1.1 Prerequisites

The Newsroom plug-in is simply a bridge between the Content Engine and a Escenic broadcast system in which newsroom management functionality is handled by the newsroom system. In such a system, Viz Content Pilot's newsroom component is displayed inside the newsroom client program and provides newsroom users with access to video and graphics content managed by the Escenic broadcast system. For information about these system, see:

- Your newsroom documentation
- The Viz Content Pilot documentation

2 Using Newsroom

The Newsroom plug-in operates in the background and does not change the way you use either the newsroom or Content Engine components in any way. Once it is configured correctly, any changes users make to certain stories in the newsroom are automatically copied to related Escenic publications.

3 Installation

The following preconditions must be met before you can install the Newsroom trunk-SNAPSHOT plug-in:

- The Content Engine and Escenic assembly tool have been installed as described in the **Escenic Content Engine Installation Guide** and are in working order.
- You have the required plug-in distribution file **`newsroom-trunk-SNAPSHOT.zip`**.
- You have access to a correctly configured and integrated broadcast system involving the following components:
  - Viz Media Engine
  - Viz Content Pilot
  - Avid iNEWS or AP Enps

3.1 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**. *escenic-home* is used to refer to the **`/opt/escenic`** folder under which both the Content Engine 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 **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:

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

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.

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.

3.2 Newsroom Installation

Installing Newsroom involves the following steps:

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

2. Download the Newsroom distribution from the Escenic Technet web site (http://technet.escenic.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 http://user:password@technet.escenic.com/downloads/release/54/newsroom-trunk-SNAPSHOT.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 Newsroom distribution file:

   ```
   $ cd /opt/escenic/engine/plugins
   $ unzip /mnt/download/newsroom-trunk-SNAPSHOT.zip
   ```

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

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

6. Run the **`ece`** script to re-assemble your Content Engine 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 the Newsroom plug-in, 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**. It only really makes sense to run the Newsroom plug-in on **editorial host**s.

8. On each **engine-host** on which you wish to run the Newsroom plug-in, deploy the EAR file and restart the Content Engine by entering:

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

   It only really makes sense to run the Newsroom plug-in on **editorial host**s.

3.3 Verify the Installation

To verify the status of the Newsroom plug-in, open the Escenic Admin web application (usually located at **`http://`***server*`/escenic-admin`) and click on **View installed plug-ins**. 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.

4 Configuration

Below we describe how to configure your Newsroom plug-in against both iNews and Enps. However in most situations you will only configure one, either iNews or Enps.

4.1 Configure iNEWS integration

In order to be able to use the iNews integration of the Newsroom after installing it (see chapter 3) you must:

1. Create a configuration for the **poller** that watches for changes in the iNEWS system.
2. Add iNEWS content types to your publication **`content-type`** resource(s).
3. Add content types for any video or online graphics embedded in the imported iNEWS stories.

4.1.1 Configure FTP Pollers

The Newsroom plug-in imports content items from iNEWS via n FTP server maintained by the iNEWS server. Stories are published on this server as XML files. The stories are organized in a folder structure that reflects the iNEWS **queue** structure.

In order to import stories from iNEWS you must:

1. Get all the information you need to be able to log in to the iNEWS FTP server (host name, user name and password)
2. Decide which iNEWS queues you want to import stories from
3. Configure a poller object to poll the relevant folders on the FTP server. The poller will then monitor these folders for changes and when a change occurs, download all new or changed files and use them to create or update corresponding content items in a particular publication.

To configure an FTP poller:

1. Decide on a name for your poller (for example **`MyINewsPoller`**)
2. Create a configuration file for your poller by copying **`/opt/escenic/engine/plugins/newsroom/misc/siteconfig/com/escenic/newsroom/ExamplePoller.properties`** to your common configuration layer. For example:

   ```
   $ mkdir -p /etc/escenic/engine/common/com/escenic/newsroom/
   $ cp /opt/escenic/engine/plugins/newsroom/misc/siteconfig/com/escenic/newsroom/
   ExamplePoller.properties \
     /etc/escenic/engine/common/com/escenic/newsroom/MyINewsPoller.properties
   ```

3. Open your configuration file for editing and add the required information (see section 4.1.1.1).
4. Open *configuration-root*`/Initial.properties` for editing and add a service entry for your poller, for example:

   ```
   service.1.00-inews-poller=/com/escenic/newsroom/MyINewsPoller
   ```

5. Restart the Content Engine.

6. Start a browser and go to **`http://`***host-name*`/escenic-admin/do/services/display`** to verify that the **`MyINewsPoller`** is running. If the service is not correctly configured you will see an error message describing the problem.

An iNEWS FTP poller imports stories to one section of one publication only. If you want to import iNEWS stories to several publications or to several sections of a publication, then you must configure one poller for each publication/section combination.

4.1.1.1 Editing the Configuration File

The following entries at the top of the **`.properties`** file you have copied should not be modified:

```
$class=com.escenic.newsroom.INewsFTPPoller
alarmManager=/neo/io/managers/AlarmManager
resourceLockManager=/io/api/DatabaseResourceLockManager
validateInterval=5
nsmlFormat=3.1nsml
objectLoader=/io/api/ObjectLoader
objectFactory=/io/api/ObjectFactory
typeManager=/io/api/TypeManager
articleTypeManager=/neo/io/managers/TypeManager
mediaHandler.graphics=./GraphicsHandler
mediaHandler.video=./VideoHandler
mediaHandler.image=./ImageHandler
executorService=/com/escenic/newsroom/Executor
componentBus=/neo/io/services/ComponentBus
connectionTimeout=60
soTimeout=120
usePassiveMode=true
```

You may, however, change or enter values for all the remaining properties, and in several cases values are required. The remaining properties are:

`delayBetweenExecutions` **(required)**
The poll interval, in seconds.

`initialDelay` **(required)**
The period of time the poller waits after start-up before starting to check the iNEWS FTP server, specified in seconds.

`serviceName` **(required)**
A name for the Content Engine service that will perform the polling. This name must be unique.

`hostName` **(required)**
The host name or IP address of the iNEWS FTP server.

`userName` **(required)**
The user name required to access the iNEWS FTP server.

`password` **(required)**
The password required to access the iNEWS FTP server.

`queues` **(required)**
A comma-separated list of the iNEWS queues from which stories are to be imported. For example:

```
queues=SHOW.ONLINE.BREAKINGNEWS,SHOW.ONLINE.SPORTSRESULTS
```

`publicationName` (required)
The name of the Escenic publication to which the selected stories are to be imported.

`contentType` (required)
The name of the Escenic content type to be used for imported iNEWS stories. This content type must be specially configured to handle iNEWS stories (see section 4.1.2).

`mediaContentType.video`
The name of the Escenic content type to be used for any video content found in imported stories (see section 4.3.2). This property is optional. If it is not specified the content type will be auto detected.

`mediaContentType.graphics`
The name of the Escenic content type to be used for any Escenic online graphics found in imported stories (see section 4.3.3). This property is optional. If it is not specified the content type will be auto detected.

`mediaContentType.image`
The name of the Escenic content type to be used for any image content found in imported stories (see section 4.3.4). This property is optional. If it is not specified the content type will be auto detected.

`relationType.video`
The name of the Escenic relation type to be used for any video content found in imported stories (see section 4.3.1). This property is required if imported stories are to include video content.

`relationType.graphics`
The name of the Escenic relation type to be used for any Escenic online graphics found in imported stories (see section 4.3.1). This property is required if imported stories are to include online graphics.

`relationType.image`
The name of the Escenic relation type to be used for any image content found in imported stories (see section 4.3.1). This property is required if imported stories are to include images.

`VMEHostName`
The host name of the VME server from which images are to be retrieved. This property is required if the stories imported from iNEWS may contain images. (It is not, however needed if imported stories will only contain video and/or online graphics.)

`VMEUserName`
The user name to be used for accessing the VME server. This property is required if the stories imported from iNEWS may contain images. (It is not, however needed if imported stories will only contain video and/or online graphics.)

`VMEPassword`
The password to be used for accessing the VME server. This property is required if the stories imported from iNEWS may contain images. (It is not, however needed if imported stories will only contain video and/or online graphics.)

`state` (required)
The default Content Engine state to be applied to newly-created content items. Any of the standard values (**`draft`**, **`approved`**, **`submitted`**, **`published`** or **`deleted`**) may be used. Which one you choose will depend on how you want to integrate iNEWS content into your Escenic workflow.

`source` (required)
A string that identifies the source of the imported content items: for example, the name of the iNEWS system. This becomes the content item's **`source`** property, which together with a

content item's **source-id** property constitutes a unique identifier. The **source-id** property of a content item imported from an iNEWS system is taken from the iNEWS story's **storyid** field. An iNEWS **storyid** is formed from three components. Two are static and form a unique ID for the story within the iNEWS system - these are imported as a content item's **source-id**. The third is dynamic, changing every time the story is updated, and is imported into a content item's **storyid** field (see section 4.1.2).

homeSectionId (required)
The ID of the section to which all content items imported by this poller are to belong.

inboxName
The name of an inbox in the home section to which imported content items are to be added. This property is optional. If it is not specified then imported content items are not added to an inbox. If it is uncommented but no value specified, like this:

```
inboxName=
```

then content items are added to the default inbox (**Inbox**).

listNames
A comma-separated list of list names. The specified lists must belong to the specified home section. This property is option. If it is not specified then imported content items are not added to any lists.

4.1.2 Adding an iNEWS Content Type

iNEWS stories (like Escenic content items) have a customized internal structure consisting of fields such as **title**, **body**, **slug** and so on. Converting an iNEWS story into an Escenic content item therefore involves:

- Selecting the iNEWS fields that contain relevant information for on-line content.
- Mapping the selected iNEWS fields to appropriate fields in an Escenic content type.

To achieve this you need to define a specially structured content type for handling iNEWS stories, and add it to your publication's **content-type** resource. The name of this content type must be specified in the **contentType** property of the FTP Poller configuration file (see section 4.1.1.1).

The newsroom content type must

- contain one **field** element which has a child **storyid** element that belongs to the **http://xmlns.escenic.com/2012/newsroom** namespace:

  ```
  <storyid xmlns="http://xmlns.escenic.com/2012/newsroom" />
  ```

  The element must be empty.

  The content item's **storyid** field will be populated with the INEWS **storyid** field. You are recommended to hide this field so that users do not inadvertently change its content. It is used to store the dynamic part of an iNEWS story's **storyid**, which changes every time a new version of the story is saved, and The Newsroom plug-in uses it when polling to determine whether or not a story has been updated in iNEWS.

The newsroom content type may also contain one or more other **field** elements that have child **field** elements belonging to the **http://xmlns.escenic.com/2012/newsroom** namespace:

```
<field xmlns="http://xmlns.escenic.com/2012/newsroom" xpath="xpath-expression"/>
```

Elements of this type must be empty, and have one attribute, **xpath**. Here is an example of a content type definition containing all required newsroom **field** elements:

```
<content-type name="newsroom"
              xmlns:newsroom="http://xmlns.escenic.com/2012/newsroom">
  <ui:title-field>title</ui:title-field>
  <panel name="main">
    <field name="title" type="basic" mime-type="text/plain">
      <newsroom:field xpath="/nsml/fields/string[@id='title']"/>
    </field>
    <field mime-type="application/xhtml+xml" type="basic" name="body">
      <newsroom:field xpath="/nsml/body"/>
    </field>
    <field mime-type="text/plain" type="basic" name="storyid">
      <ui:hidden/>
      <newsroom:storyid />
    </field>
    <field mime-type="text/plain" type="basic" name="summary">
    </field>
  </panel>
  ...
</content-type>
```

A **newsroom:field** element's **xpath** attribute must contain a valid **XPath** expression that selects the iNEWS field you want to appear in its parent **field**. In the example above, the content item's **title** field will receive content from the iNEWS **title** field (selected by the XPath expression **/nsml/fields/string[@id='title']**). In addition to this, the content item's **body** field will receive content from the iNEWS **body** field (selected by the XPath expression **/nsml/body**)

In order to be able to configure these fields correctly, you need to know:

- How to use XPath (see http://www.w3.org/TR/xpath/)
- The structure of the XML files published on the iNEWS FTP server. For information about this, consult the appropriate iNEWS documentation.

The content type may also contain other fields that do not have child **newsroom:field** elements (like the **summary** field in the example above). Such fields will simply be left empty and can, for example, be filled after import by Content Studio users.

4.2 Configure Enps integration

The Newsroom communicates with Enps using the MOS Protocol. Each time an event is occurred in Enps, it sends appropriate messages according to the protocol to all registered MOS devices. This plugin enables your Escenic Content Engine installation to function as a MOS device.

The Newsroom provides facility to follow activities related to rundowns in Enps by processing those event messages. The activities include creation and modification of stories; insertion, removal, reordering of stories within a rundown and creation, modification and removal of rundowns, etc. To help the plugin to achieve this you need to configure the following.

In order to import stories from Enps you must:

1. Register your server as a **MOS device** in Enps.

2. Create a new rundown in Enps and tell it to send **roStorySend** messages when a story is created or updated.
3. Enable the **Device**
4. Create a **StoryCreator**
5. Add Enps content type to your publications **content-type** resource

To follow rundowns you need to:

1. Create a configuration for each of the Enps server you want to follow
2. Register those configurations to the rundown manager

In order to send stories to Enps you must:

1. Create a new rundown in Enps and send **roReqStoryAction** messages to Enps when an article is created in ECE.
2. Enable the **Device**
3. Create a **StoryCreator**
4. Add Enps content type to your publications **content-type** resource

> Caution should be taken while configuring rundown in **Enps**. The rundown where we want to create stories for contents created in **Escenic content engine** should not be configured to create contents in **Escenic Content Engine** for those stories. This is for avoiding potential looping between **ECE** and **Enps**.

4.2.1 Configure the Device

Open **`com/escenic/mos/Device.properties`** in an editor and add

```
serviceEnabled=true
```

and

```
bindAddress=host name
```

Where *host name* is the name or ip address of **this** host.

4.2.2 Configure a story creator

A **`story creator`** is the component creating the actual article.

To configure a story creator:

1. Decide on a name for the story creator (for example **`MyStoryCreator`**)
2. Create a configuration file by copying **`/opt/escenic/engine/plugins/newsroom/misc/siteconfig/com/escenic/mos/handlers/ExampleStoryCreator.properties`** to your common configuration layer. For example:

   ```
   $ mkdir -p /etc/escenic/engine/common/com/escenic/mos/handlers
   $ cp /opt/escenic/engine/plugins/newsroom/misc/siteconfig/com/escenic/mos/
   handlers/ExampleStoryCreator.properties \
     /etc/escenic/engine/common/com/escenic/mos/handlers/MyStoryCreator.properties
   ```
3. Open your configuration file for editing and add the required information (see section 4.2.2.1).

4. Open *configuration-root*`/com/escenic/mos/handlers/roStorySend.properties` for editing and add a reference to your story creator, for example:

   ```
   storyCreators=/com/escenic/mos/handlers/MyStoryCreator
   ```

5. Restart the Content Engine.
6. Start a browser and go to `http://`*host-name*`/escenic-admin/do/services/display` to verify that the `MyStoryCreator` is running. If the service is not correctly configured you will see an error message describing the problem.

A story creator imports stories to one section of one publication only. If you want to import Enps stories to several publications or to several sections of a publication, then you must configure one story creator for each publication/section combination. Story creator is also responsible to generate **roReqStoryAction** message.

4.2.2.1 Editing the Configuration File

The following entries at the top of the `.properties` file you have copied should not be modified:

```
$class=com.escenic.mos.handler.MosStoryCreator
objectLoader=/io/api/ObjectLoader
objectFactory=/io/api/ObjectFactory
typeManager=/io/api/TypeManager
articleTypeManager=/neo/io/managers/TypeManager
mediaHandler.graphics=/com/escenic/newsroom/GraphicsHandler
mediaHandler.video=/com/escenic/newsroom/VideoHandler
mediaHandler.image=/com/escenic/newsroom/ImageHandler
```

You may, however, change or enter values for all the remaining properties, and in several cases values are required. The remaining properties are:

`publicationName` (required)
The name of the Escenic publication to which the selected stories are to be imported.

`contentType` (required)
The name of the Escenic content type to be used for imported Enps stories. This content type must be specially configured to handle Enps stories (see section 4.2.3).

`state` (required)
The default Content Engine state to be applied to newly-created content items. Any of the standard values (`draft`, `approved`, `submitted`, `published` or `deleted`) may be used. Which one you choose will depend on how you want to integrate Enps content into your Escenic workflow.

`source` (required)
A string that identifies the source of the imported content items: for example, the name of the Enps system. This becomes the content item's `source` property, which together with a content item's `source-id` property constitutes a unique identifier. The `source-id` property of a content item imported from an Enps system is taken from the Enps story's `storyid` field.

`homeSectionId` (required)
The ID of the section to which all content items imported by this poller are to belong.

inboxName
The name of an inbox in the home section to which imported content items are to be added. This property is optional. If it is not specified then imported content items are not added to an inbox. If it is uncommented but no value specified, like this:

```
inboxName=
```

then content items are added to the default inbox (**Inbox**).

listNames
A comma-separated list of list names. The specified lists must belong to the specified home section. This property is option. If it is not specified then imported content items are not added to any lists.

XPath
An optional XPath expression that makes it possible to validate the MOS document before creating a story of it. This could for instance be used to only import stories from a given rundown, only import stories with a given content, etc.

mediaContentType.video
The name of the Escenic content type to be used for any video content found in imported stories (see section 4.3.2). This property is optional. If it is not specified the content type will be auto detected.

mediaContentType.graphics
The name of the Escenic content type to be used for any Escenic online graphics found in imported stories (see section 4.3.3). This property is optional. If it is not specified the content type will be auto detected.

mediaContentType.image
The name of the Escenic content type to be used for any image content found in imported stories (see section 4.3.4). This property is optional. If it is not specified the content type will be auto detected.

relationType.video
The name of the Escenic relation type to be used for any video content found in imported stories (see section 4.3.1). This property is required if imported stories are to include video content.

relationType.graphics
The name of the Escenic relation type to be used for any Escenic online graphics found in imported stories (see section 4.3.1). This property is required if imported stories are to include online graphics.

relationType.image
The name of the Escenic relation type to be used for any image content found in imported stories (see section 4.3.1). This property is required if imported stories are to include images.

VMEHostName
The host name of the VME server from which images are to be retrieved. This property is required if the stories imported from Enps may contain images. (It is not, however needed if imported stories will only contain video and/or online graphics.)

VMEUserName
The user name to be used for accessing the VME server. This property is required if the stories imported from Enps may contain images. (It is not, however needed if imported stories will only contain video and/or online graphics.)

VMEPassword
The password to be used for accessing the VME server. This property is required if the stories imported from Enps may contain images. (It is not, however needed if imported stories will only contain video and/or online graphics.)

roId
The rundown of the Enps where story will be created for contents in ECE

4.2.3 Adding an Enps Content Type

Enps stories (like Escenic content items) have a customized internal structure consisting of fields such as **title**, **body**, **storySlug** and so on. Converting an Enps story into an Escenic content item therefore involves:

- Selecting the Enps fields that contain relevant information for on-line content.
- Mapping the selected Enps fields to appropriate fields in an Escenic content type.

To achieve this you need to define a specially structured content type for handling Enps stories, and add it to your publication's **content-type** resource. The name of this content type must be specified in the **contentType** property of the story creator configuration file (see section 4.2.2.1).

The enps content type must

- contain one **field** element which has a child **storyid** element that belongs to the **http://xmlns.escenic.com/2012/newsroom** namespace:

  ```
  <storyid xmlns="http://xmlns.escenic.com/2012/newsroom" />
  ```

  The element must be empty.

  The content item's **storyid** field will be populated with the Enps **storyid** field. You are recommended to hide this field so that users do not inadvertently change its content.

The newsroom content type may also contain one or more other **field** elements that have child **field** elements belonging to the **http://xmlns.escenic.com/2012/newsroom** namespace:

```
<field xmlns="http://xmlns.escenic.com/2012/newsroom" name="storySlug|storyBody"
 xpath="xpath-expression"/>
```

Elements of this type must be empty, and have one attribute, **xpath**. Here is an example of a content type definition containing all required newsroom **field** elements:

```
  <content-type name="newsroom"
                xmlns:newsroom="http://xmlns.escenic.com/2012/newsroom">
    <ui:title-field>title</ui:title-field>
    <panel name="main">
      <field name="title" type="basic" mime-type="text/plain">
        <newsroom:field name="storySlug" xpath="/mos/roStorySend/storySlug"/>
      </field>
      <field mime-type="application/xhtml+xml" type="basic" name="body">
        <newsroom:field name="storyBody" xpath="/mos/roStorySend/storyBody"/>
      </field>
      <field mime-type="text/plain" type="basic" name="storyid">
        <ui:hidden/>
        <newsroom:storyid />
      </field>
      <field mime-type="text/plain" type="basic" name="summary">
```

```
      </field>
    </panel>
    ...
  </content-type>
```

A **newsroom:field** element's **xpath** attribute must contain a valid **XPath** expression that selects the Enps field you want to appear in its parent **field**. In the example above, the content item's **title** field will receive content from the Enps **storySlug** field (selected by the XPath expression **/mos/roStorySend/storySlug**). In addition to this, the content item's **body** field will receive content from the Enps **storyBody** field (selected by the XPath expression **/mos/roStorySend/storyBody**)

In order to be able to configure these fields correctly, you need to know:

- How to use XPath (see http://www.w3.org/TR/xpath/)
- The structure of the **roStorySend** message created by Enps. For information about this, consult the appropriate Enps and MOS Protocol (see http://www.mosprotocol.com/MOS%20Files/MOS%20v28-510.htm) documentation.
- The structure of the **roReqStoryAction** message that Enps accepts. For information about this, consult the appropriate Enps and MOS Protocol (see http://http://www.mosprotocol.com/MOS%20Files/MOSv283-548.htm) documentation.

The content type may also contain other fields that do not have child **newsroom:field** elements (like the **summary** field in the example above). Such fields will simply be left empty and can, for example, be filled after import by Content Studio users.

4.2.4 Create a Newsroom Configuration

A **Newsroom Configuration** contains the information required to communicate effectively to the newsroom server, i.e. Enps.

To configure a newsroom configuration you need to:

1. Decide a name for a newsroom server, i.e. MyEnpsConfiguration
2. Create a configuration file by copying **/opt/escenic/engine/plugins/newsroom/misc/siteconfig/com/escenic/mos/ExampleConfiguration.properties** to your common configuration layer. For example:

   ```
   $ mkdir -p /etc/escenic/engine/common/com/escenic/mos/
   $ cp /opt/escenic/engine/plugins/newsroom/misc/siteconfig/com/escenic/mos/
   ExampleConfiguration.properties \
                /etc/escenic/engine/common/com/escenic/mos/
   MyEnpsConfiguration.properties
   ```

3. Open the configuration file for editing and modify all the properties according to your Enps installation except the **$class**.

The properties you need to modify are:

mosID
: The ID of the mos device you have configured at the Enps server for this ECE installation

ncsID
: The ID of the Enps server

userName
The userName to use when create story in the Enps server

ncsIPAddress
The IP address of the Enps server

lowPort
The lower port of the Enps server to connect

highPort
The upper port of the Enps server to connect

mosRev
MOS protocol version number(2.8.3|2.8.4) which is supported by the Enps

4.2.5 Configure the Rundown Manager

Open the **/etc/escenic/engine/common/com/escenic/mos/ROManager.properties** for editing (if not exists, then create one) and modify or add the **newsroomConfiguration** property. This property a configuration of **Newsroom Configuration**.

An example configuration:

```
newsroomConfiguration=/com/escenic/mos/MyEnpsConfiguration
```

4.3 Enabling Video, Online Graphics and Image

4.3.1 Configure relation types

If the newsroom stories to be imported to a publication will include video, online graphics or images, then the content type definition must include relations for them. For example:

```
<content-type name="newsroom"
        xmlns:newsroom="http://xmlns.escenic.com/2012/newsroom">
        ...
            <relation-type name="inews-video">
          </relation-type>
          <relation-type name="inews-online-graphics">
          </relation-type>
          <relation-type name="inews-image">
          </relation-type>
            ...
        </content-type>
```

The names of these relations must match the names specified in the **relationType.video** and **relationType.graphics** properties of the FTP Poller or story creator configuration file (see section 4.1.1.1 or section 4.2.2.1).

4.3.2 Adding a Video Content Type

If the newsroom stories to be imported to a publication will include video, then the publication's **content-type** resource must contain a suitable content type. For example:

```
<content-type name="video">
          <parameter name="com.vizrt.video" value="true"/>
          <ui:icon>graphic</ui:icon>
          <ui:label>External video</ui:label>
          <ui:title-field>title</ui:title-field>
          <panel name="main">
          <field name="title" type="basic"/>
              <field name="video" type="basic" mime-type="application/json">
            <video xmlns="http://xmlns.escenic.com/2010/video" enabled="true"/>
            </field>
            </panel>
          </content-type>
```

The important points to note are highlighted in the example:

A video content type must at least have the following:

- The content type must have a **parameter** element called **com.vizrt.video** that is set to **true**
- It must also have a **basic** field with:
  - **mime-type** set to **application/json**
  - a **video** sub-element with a **value** attribute set to **true**. This element must belong to the namespace **http://xmlns.escenic.com/2010/video**.

A video content type that is to be used by the Newsroom plug-in must **not** contain any binary fields.

4.3.3 Adding an Online Graphics Content Type

If the newsroom stories to be imported to a publication will include Escenic online graphics, then the publication's content-type resource must contain a suitable content type. The following example shows such a content type definition:

```
  <content-type name="online-graphic">
          <ui:icon>image</ui:icon>
          <ui:label>On-line Graphic</ui:label>
          <ui:title-field>title</ui:title-field>
          <ui:decorator
class="com.escenic.onlinegraphics.presentation.PreviewServerDecorator"/>
          <panel name="default">
          <field mime-type="text/plain" name="title" type="basic">
          <ui:label>Title</ui:label>
          <ui:description>The title of the article</ui:description>
          <constraints>
          <required>true</required>
          </constraints>
          </field>
                <field mime-type="application/json" name="graphics" type="basic">
                  <ui:label>Graphic</ui:label>
                  <graphics xmlns="http://xmlns.escenic.com/2010/graphics"/>
            </field>
          </panel>
          <summary>
          <ui:label>Content Summary</ui:label>
          <field mime-type="text/plain" name="title" type="basic">
          <ui:label>Title</ui:label>
          </field>
```

```
            </summary>
            </content-type>
```

The important points to note are highlighted in the example:

- The **content-type** element must contain a child **ui:decorator** element specifying the **PreviewServerDecorator** class.
- The field that contains the actual graphic:
  - must have its **mime-type** attribute set to **application/json**
  - must have a child element called **graphics** that belongs to the namespace **http://xmlns.escenic.com/2010/graphics**.

A content type may contain **only one** online graphics field.

4.3.4 Adding an Image Content Type

If the newsroom stories to be imported to a publication will include image, then the publication's **content-type** resource must contain a suitable content type. For example:

```
<content-type name="image">
          <ui:label>Picture</ui:label>
          <ui:icon>image</ui:icon>
          <ui:title-field>name</ui:title-field>
          <panel name="main">
          <field mime-type="text/plain" type="basic" name="name"/>
              <field name="binary" type="link">
            <relation>com.escenic.edit-media</relation>
            <constraints>
            <mime-type>image/jpeg</mime-type>
            <mime-type>image/png</mime-type>
            </constraints>
            </field>
          </panel>
          </content-type>
```

The important points to note are highlighted in the example:

A image content type must at least have the following:

- It must have a **link** field with:
  - **relation** set to **com.escenic.edit-media**
  - at least one **mime-type** constraint that starts with the string **image/**.

4.4 Tips & tricks

4.4.1 XPATH tips

If you need to check more than one file for content in the configuration you can use the **pipe** character (|) to do so. By example like this:

```
<newsroom:field xpath="/mos/roStorySend/storyBody | /nsml/body"/>
```

This will first search for a **`/mos/roStorySend/storyBody`** and then for **`/nsml/body`** if the first one was not found.