Cxense Semantic

# Plug-in Guide

1.4.2.166554

escenic

A CCI COMPANY

# Table of Contents

# 1 Introduction

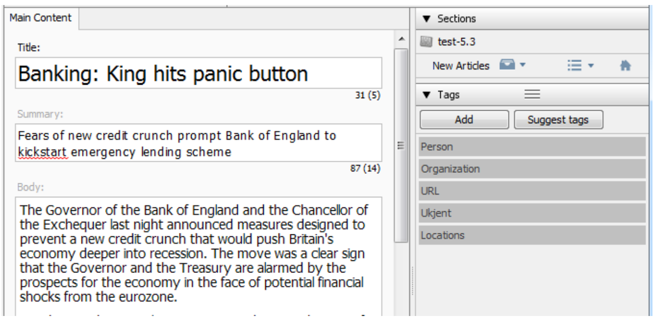
The Cxense Semantic plug-in for Escenic Content Engine provides an interface to the Cxense entity tagging service.

The Cxense entity tagging service searches submitted content for references to various subjects and returns tags associated with the subject references it finds. The returned tags are weighted for relevance.

The Cxense Semantic plug-in enables the Content Engine to:

- Submit content items to Cxense for analysis and tagging
- Map the returned Cxense tags to corresponding Escenic tags if they exist, or else create new tags
- Convert Cxense tag relevance weightings to Escenic tag relevance values

The Cxense Semantic plug-in also adds a small extension to the Content Studio user interface to provide user access to the service. This extension consists of a **Suggest tags** button, which appears next to the **Add** button in the **Tags** section of the attributes panel:



Banking: King hits panic button

Fears of new credit crunch prompt Bank of England to kickstart emergency lending scheme

The Governor of the Bank of England and the Chancellor of the Exchequer last night announced measures designed to prevent a new credit crunch that would push Britain's economy deeper into recession. The move was a clear sign that the Governor and the Treasury are alarmed by the prospects for the economy in the face of potential financial shocks from the eurozone.

Clicking on the **Suggest tags** button submits the text of the current content item to Cxense, and adds the returned tags to the content item.

## 1.1 Automated Tagging

A changelog daemon that can perform automated tagging is included with the Cxense Semantic plug-in. This daemon runs in the background and automatically submits untagged content items to Cxense for tagging. You can configure it to only submit content items of specified types, and only when they are in specified states.

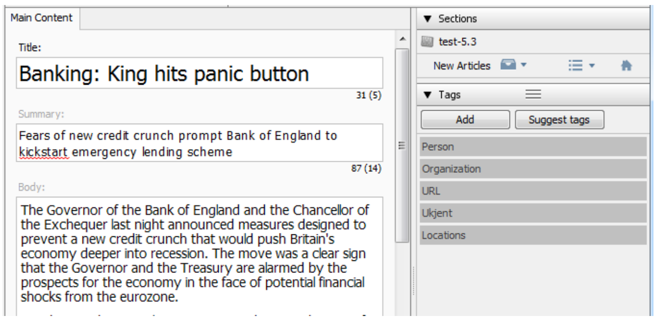
The first time the changelog daemon is run, it will work through all content items in the Content Engine looking for items to tag. Once it has "caught up", it will simply submit new and modified content items of the specified types/states.

## 1.2 Prerequisites

The only pre-requisite for using the Cxense Semantic plug-in is a Cxense account. In order for Cxense to work well with your publications you have to train it after setting up your account.

# 2 Using the Cxense Semantic Plug-in

Once the Cxense Semantic plug-in is correctly installed and configured, a new **Suggest tags** button may appear in the **Tags** section of the attributes panel when editing content items. It may not necessarily appear with all content items - your publication designers can control which kinds of content items make use of the tagging service.



Title:

Banking: King hits panic button

31 (5)

Summary:

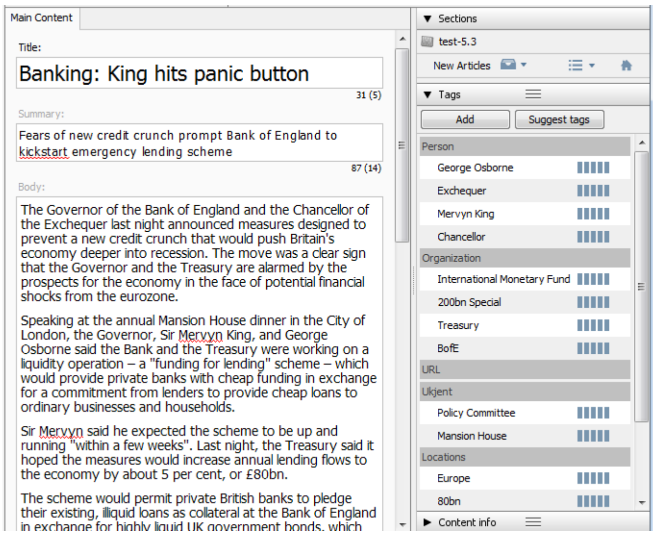
Fears of new credit crunch prompt Bank of England to kickstart emergency lending scheme

87 (14)

Body:

The Governor of the Bank of England and the Chancellor of the Exchequer last night announced measures designed to prevent a new credit crunch that would push Britain's economy deeper into recession. The move was a clear sign that the Governor and the Treasury are alarmed by the prospects for the economy in the face of potential financial shocks from the eurozone.

If this button is displayed, then you can auto-tag the content item by clicking on **Suggest tags**. If Cxense returns any tags, then they are added to the content item as suggestions:



Title:

Banking: King hits panic button

31 (5)

Summary:

Fears of new credit crunch prompt Bank of England to kickstart emergency lending scheme

87 (14)

Body:

The Governor of the Bank of England and the Chancellor of the Exchequer last night announced measures designed to prevent a new credit crunch that would push Britain's economy deeper into recession. The move was a clear sign that the Governor and the Treasury are alarmed by the prospects for the economy in the face of potential financial shocks from the eurozone.

Speaking at the annual Mansion House dinner in the City of London, the Governor, Sir Mervyn King, and George Osborne said the Bank and the Treasury were working on a liquidity operation – a "funding for lending" scheme – which would provide private banks with cheap funding in exchange for a commitment from lenders to provide cheap loans to ordinary businesses and households.

Sir Mervyn said he expected the scheme to be up and running "within a few weeks". Last night, the Treasury said it hoped the measures would increase annual lending flows to the economy by about 5 per cent, or £80bn.

The scheme would permit private British banks to pledge their existing, illiquid loans as collateral at the Bank of England in exchange for highly liquid UK government bonds, which

If Cxense can't find any references to subjects that it knows about, then it will not return any tags and no suggestions are made.

If you are satisfied with the tags suggested by Cxense, then you can just save the content item and continue working. If you are not satisfied with the suggestions, you can:

- Remove suggested tags
- Add other tags not suggested by Cxense
- Change the relevance values set by Cxense

## 2.1 Managing Tags

Cxense can generate different categories of tags, and the Cxense Semantic plug-in can be configured to assign these categories to corresponding tag collections in your publication. If a tag suggestion returned by Cxense matches an existing tag or tag alias in your tag collection, then that tag is used. If your collection does not contain a matching tag, then the plug-in creates one.

Escenic tag collections can have a tree structure, but the tags returned by Cxense are not structured in any way. If a returned tag does not match an existing tag in your structure, then it will be created as a free-standing tag at the top level of your structure. Once it has been created, however, you can move it to the required location in your tag structure using the **Manage Tags** dialog (see The Manage Tags Dialog).

Tags returned by Cxense are mapped on to tags in your target tag collection by string matching. This means that new tags will sometimes be created unnecessarily due to the use of synonyms or spelling variations. When this occurs you can "tidy up" by using the **Manage Tags** dialog's **Merge** function (see Merging Tags) to merge the new tags with existing ones.

# 3 Installation

The following preconditions must be met before you can install the Cxense Semantic 1.4.2.166554 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 **`semantic-cxense-1.4.2.166554.zip`**.

## 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 Cxense Semantic Installation

Installing Cxense Semantic involves the following steps:

1. Log in as **`escenic`** on your **assembly-host**.
2. Download the Cxense Semantic 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/5.7.unstable-
VF-5970.161071/cxense-semantic-1.4.2.166554.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 Cxense Semantic distribution file:

   ```
   $ cd /opt/escenic/engine/plugins
   $ unzip /mnt/download/cxense-semantic-1.4.2.166554.zip
   ```

   This will result in the creation of an **`/opt/escenic/engine/plugins/cxense-semantic`** 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 Cxense Semantic 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 Cxense Semantic plug-in on **editorial host**s.

8. On each **engine-host** on which you wish to run the Cxense Semantic 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 Cxense Semantic plug-in on **editorial host**s.

## 3.3 Installing The Change Log Daemon

If you want to make use of the Cxense Semantic plug-in's automated tagging functionality, then as well as installing the Cxense Semantic plug-in itself, you also need to install a **change log daemon** - a program that scans the Content Engine change logs looking for changed content items, and submits them to Cxense for tagging if necessary. You can configure it to only submit certain content types, and to only submit content items in certain states.

The instructions below should provide enough information for you to install and configure the autotagging daemon, but if you want to know more about how change log daemons in general work, then take a look at the Change Log Daemon documentation.

The Autotagging change log daemon checks for new and/or modified content items and submits them to Cxense for tagging. You can configure it to only submit certain content types, and to only submit content items in certain states. To install it:

1. Download the Change Log Daemon distribution from the Content Engine downloads page, and unpack it in an appropriate location (**`/opt/escenic/semantic/autotagging`**, for example).
2. Change directory to the new folder you have created and start the daemon at least once with the following command:

   ```
   $ java -jar changelog.jar
   ```

   This creates a few required folders and configuration files.
3. Open **`config/Daemon.properties`** for editing and set the following properties:

   **`url`**
   The URI of your publication's change log. For example:

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

   Where *editorial-host-ip-address* is the host name of IP address of your **editorial-host**, and *publicationId* is your publication's ID.

   **`username`**
   The username of the Content Engine user that will be used access the change log.

   **`password`**
   The password of the above user.

The change log daemon is now configured with sufficient information to read your publication's change log. You now need to add the agent that manages the tagging of new and changed content items. To do this:

1. Change directory to the autotagging daemon folder. For example:

   ```
   $ cd /opt/escenic/semantic/autotagging/changelog-daemon
   ```
2. Copy the content of **`/opt/escenic/engine/plugin/semantic-cxense/misc/changlog/lib`** to the empty **`lib`** folder.

   ```
   $ cp /opt/escenic/engine/plugin/semantic-cxense/misc/changlog/lib/* lib
   ```
3. Create a folder tree under the empty **`classes`** folder:

   ```
   $ mkdir -p classes classes/com/escenic/daemon/
   ```
4. Copy **`/opt/escenic/engine/plugins/semantic-cxense/misc/example/SemanticAgent.properties`** to the **`classes/com/escenic/daemon/`** folder you have created:

   ```
   $ cp /opt/escenic/engine/plugins/semantic-cxense/misc/example/
   SemanticAgent.properties \
         classes/com/escenic/daemon/
   ```
5. Open the copied file for editing and set the following properties:

   **`webserviceEndpoint`**
   Set this property to the URI of your Content Engine web service:

   ```
   webserviceEndpoint=https://host:port/webservice/
   ```

   where *host* and *port* are the host name and port number of your Content Engine host.

**keepLastModified**
: You can use this property to determine whether or not the autotagging daemon updates content items' **LastModified** property when it adds tags to them:

- **true**: **LastModified** is **not** updated when tags are added by the daemon
- **false**: **LastModified is** updated when tags are added by the daemon. This is the default setting.

6. Copy **/opt/escenic/engine/plugins/semantic-cxense/misc/example/SemanticConfiguration.properties** to the same folder:

```
$ cp /opt/escenic/engine/plugins/semantic-cxense/misc/example/
SemanticConfiguration.properties \
      classes/com/escenic/daemon/
```

7. Open the copied file for editing and set the following properties:

**endpoint**
: Set this property to the URI of your semantic web service:

```
endpoint=https://host:port/webservice-extensions/semantic/webservice
```

where *host* and *port* are the host name and port number of your Content Engine host.

**state.draft, state.approved, state.submitted**
: Use these properties to specify the content type/state combinations that you want to be automatically tagged. For example:

```
state.draft=story,review
state.approved=story
state.submitted=
```

The above example specifies that new or changed **story** and **review** content items will be automatically tagged if they are in a draft state. In addition new or changed **story** content items will also be automatically tagged if they are in an approved state. No content items will be automatically tagged while they are in the submitted state.

Your tagging daemon is now fully configured and ready to use. You can start it with the following command:

```
$ java -jar changelog.jar
```

You should probably add this command to the start-up script of your server.

## 3.4 Verify the Installation

To verify the status of the Cxense Semantic 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

In order to be able to use the Cxense Semantic plug-in after installing it (see chapter 3) you must:

1. Configure the Cxense Semantic web service that submits content items to Cxense for analysis and then adds the tags suggested by Cxense
2. Define a Content Engine proxy for the CXense analytics service used by the plug-in
3. Configure the content types that are to be tagged by Cxense

## 4.1 Configure the Semantic Web Service

All the tagging operations performed by the Cxense Semantic plug-in are carried out via the semantic web service. The semantic web service is installed together with the plug-in. To configure the web service:

1. Create a configuration file by copying **`Configuration.properties`** from the plug-in distribution to your webapp configuration layer. For example:

```
$ cp /opt/escenic/engine/plugins/semantic-cxense/misc/siteconfig/webservice-
extensions/com/escenic/semantic/Configuration.properties \
  /etc/escenic/engine/webapp/webservice-extensions/com/escenic/semantic/
Configuration.properties
```

2. Open your configuration file for editing.
3. Set the following properties:

**`endPoint`**

The URL of the Cxense tagging service. For example:

```
endPoint=https://api.cxense.com/processing/linguistics/execute?
persisted=tagging-id
```

where *tagging-id* is the ID of a Cxense **persisted API URL**. In order to create such a URL you need to have set up a Cxense account. Once you have a Cxense account you can use the Cxense API to create the persisted API URLs you need, and get their IDs. For a description of how to do this, see section 4.3.

**`category.`*`name`***

You need to add one setting like this for each Cxense tag category that you want to use in your publications, where *name* is the name of the Cxense category you want to use. The value you assign to the setting must be the **scheme** of the Content Engine tag structure you want to be mapped to the specified Cxense category. For example:

```
category.organization=tag:organization@example.com,2011
```

Note that you can, if you wish, map one CXense tag category to multiple Content Engine tag structures. You must then separate the scheme names with a `;` character. For example:

```
category.organization=tag:organization@example.com,2011;tag:company@example.com,2012
```

**`minRelevance`**

A minimum relevance below which tag suggestions will be rejected. Only tags which are assigned at least this relevance by Cxense will be accepted as suggestions by the Cxense

Semantic plug-in. The specified value must be a number between 0 (no relevance) and 1 (maximum relevance). For example:

```
minRelevance=0.5
```

The Content Engine tag structures you map on to Cxense categories must exist (that is, they must be defined in the Content Engine). Tag structures are defined using the **`escenic-admin`** web application. For details, see Create a Tag Structure.

## 4.2 Define the Analytics Proxy Service

Content Studio needs to communicate with the Cxense analytics service. For security reasons, it should not communicate with the Cxense service directly, but via Content Engine proxy service. For a full description of how to configure a Content Engine proxy service, see Defining a Proxy Service.

The **`serviceMapping`** definition for this proxy service should be:

```
serviceMapping.cxense-analytics-url=https://api.cxense.com/traffic/event?
persisted=analytics-id
```

where *analytics-id* is the ID of a Cxense **persisted API URL**. In order to create such a URL you need to have set up a Cxense account. Once you have a Cxense account you can use the Cxense API to create the persisted API URLs you need, and get their IDs. For a description of how to do this, see section 4.3.

## 4.3 Creating Cxense Persisted API URLs

For general information on how to get started using the Cxense API, see API authentication. If you use the Python command line tool described in that introduction, then you can create the required URLs with the following commands:

**Tagging id**

Enter the following command:

```
python cx.py /persisted/create \
'{"path":"/processing/linguistics/execute", "request":{}, "mutable":
{"document":true, "documentId":true}}'
```

The response will look something like this:

```
{ "id": "4b47ecceb282ede72d0d4087d06dd9d135b375ab" }
```

The long string is the *tagging-id* you need to include in your semantic web service configuration (see section 4.1).

**Analytics id**

Enter the following command:

```
python cx.py /persisted/create \
'{"path":"/traffic/event","request":
{"siteId":"9222310557536506915", "start":-3600, "groups":["url"],
 "historyFields":["events","sessionStarts"], "historyBuckets":60, "fields":
["uniqueUsers", "activeTime", "sessionStarts", "sessionStops"]}
```

The response will look something like this:

```
{ "id": "a50ed5026291b67fc31e82164134633c005c8def" }
```

The long string is the *analytics-id* you need to include in your analytics service mapping definition (see section 4.2).

## 4.4 Configure Content Types

The Cxense Semantic plug-in only displays a **Suggest tags** button for content items if they belong to a content type for which auto-tagging has been enabled. To enable auto-tagging you need to add a number of elements to the content type definitions in your publication's **`content-type`** resource. In each content-type that is to be auto-tagged you must:

- Mark the fields containing taggable text
- Add a hidden field to hold the response returned by Cxense

In order to enable any kind of tagging (not just auto-tagging), a content type definition must contain **`ui:tag-scheme`** elements specifying the tag collections to be used (see Controlling Tagging). In order for auto-tagging to work, these tag collections must be the ones you have associated with the Cxense tag categories.

### 4.4.1 Marking Taggable Fields

To mark a field as taggable you must add a child **`field`** element that belongs to the namespace **`http://xmlns.escenic.com/2011/semantic-cxense`**. This is an empty element with one attribute, **`name`**. **`name`** may contain one of the following values:

**`title`**
: The contents of this field will be submitted for tagging in Cxense's "'title" field. Text submitted in this field is treated as part of the title of the submitted item. Tags that appear this field are therefore likely to have higher relevance. Normally you should only use **`title`** for your title field.

**`body`**
: The contents of this field will be submitted for tagging in Cxense's "'body" field. Text submitted in this field is treated as ordinary content. Normally you should use **`body`** for all fields (except your title field) that you want to be tagged.

For example:

```
<field mime-type="application/xhtml+xml" type="basic" name="body">
  <ui:label>Body</ui:label>
  <ui:description>The body text of the article.</ui:description>
  <field xmlns="http://xmlns.escenic.com/2011/semantic-cxense" name="body"/>
</field>
```

### 4.4.2 Adding a Response Field

A response field is required in each content type to hold responses returned from Cxense. The response field must:

- Be a basic field with the MIME type **`application/json`**.

- Contain a child **`field`** element belonging to the namespace **`http://xmlns.escenic.com/2011/semantic`**. This is an empty element with one attribute, **`name`**. **`name`** must be set to the value **`response`**.

It is also recommended that you hide this field by adding a child **`ui:hidden`** element.

For example:

```
<field mime-type="application/json" type="basic" name="cxense-response">
  <ui:label>Cxense response</ui:label>
  <ui:description>The JSON response from cxense.</ui:description>
  <ui:hidden/>
  <semantic:field name="response"/>
</field>
```

### 4.4.3 Example Content Type

The following example shows a complete minimal content type that is configured for auto-tagging.

```
<content-types xmlns="http://xmlns.escenic.com/2008/content-type"
               xmlns:ui="http://xmlns.escenic.com/2008/interface-hints"
               version="4">
  <content-type name="news-cxense"
                xmlns:semantic="http://xmlns.escenic.com/2011/semantic"
                xmlns:cxense="http://xmlns.escenic.com/2011/semantic-cxense">
    <ui:label>Cxense Story</ui:label>
    <ui:description>A news story auto tagged by cxense</ui:description>
    <ui:icon>news</ui:icon>
    <ui:title-field>title</ui:title-field>
    <ui:tag-scheme>tag:person@example.com,2011</ui:tag-scheme>
    <ui:tag-scheme>tag:organization@example.com,2011</ui:tag-scheme>
    <ui:tag-scheme>tag:location@example.com,2011</ui:tag-scheme>
    <ui:tag-scheme>tag:unknown@example.com,2011</ui:tag-scheme>
    <ui:tag-scheme>tag:url@example.com,2011</ui:tag-scheme>
    <panel name="main">
      <ui:label>Main Content</ui:label>
      <ui:description>The main content fields</ui:description>
      <field name="title" type="basic" mime-type="text/plain">
        <ui:label>Title</ui:label>
        <constraints>
          <required>true</required>
        </constraints>
        <cxense:field name="title"/>
      </field>
      <field mime-type="text/plain" type="basic" name="summary">
        <ui:label>Summary</ui:label>
        <cxense:field name="body"/>
      </field>
      <field mime-type="application/xhtml+xml" type="basic" name="body">
        <ui:label>Body</ui:label>
        <cxense:field name="body"/>
      </field>
      <field mime-type="application/json" type="basic" name="cxense-response">
        <ui:hidden/>
        <semantic:field name="response"/>
      </field>
    </panel>
    <summary>
```

```
      <ui:label>Content Summary</ui:label>
      <field name="title" type="basic" mime-type="text/plain"/>
      <field name="summary" type="basic" mime-type="text/plain"/>
    </summary>
  </content-type>
</content-types>
```