Saplo Semantic

Plug-in Guide

1.3.1.159245

escenic

A CCI COMPANY

Table of Contents

1 Introduction 3

1.1 Prerequisites 3

2 Using the Saplo Semantic Plug-in 4

2.1 Managing Tags 5

3 Installation 6

3.1 Conventions 6

3.2 Saplo Semantic Installation 6

3.3 Saplo Semantic Feedback Changelog Daemon Installation 7

3.4 Saplo Semantic Ontology Changelog Daemon Installation 9

3.5 Verify the Installation 10

4 Configuration 11

4.1 Configure Saplo Log-in Credentials 11

4.2 Configure Content Studio 11

4.3 Configure Content Types 12

4.3.1 Marking Taggable Fields 13

4.3.2 Adding a Response Field 13

4.3.3 Example Content Type 13

1 Introduction

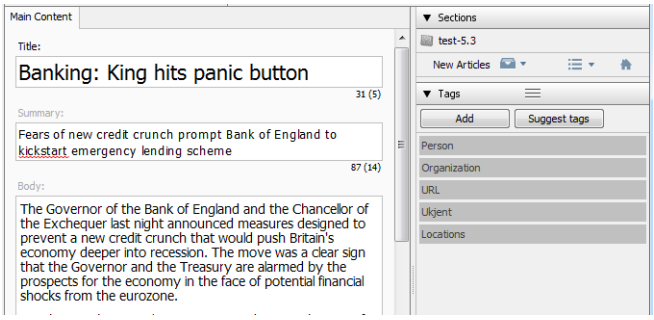
The Saplo Semantic plug-in for Escenic Content Engine provides an interface to the Saplo entity tagging service.

The Saplo entity tagging service searches submitted content for references to various subjects (or **entities** in Saplo terminology) and returns tags associated with the entity references it finds. The returned tags are weighted for relevance. Currently, Saplo recognizes the following types of entities: persons, organizations, locations, URLs and other.

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

- Submit content items to Saplo for analysis and tagging
- Map the returned Saplo tags to corresponding Escenic tags if they exist, or else create new tags
- Convert Saplo tag relevance weightings to Escenic tag relevance values
- Submit any manual changes made to tags or relevance back to Saplo. Saplo uses this feedback to customize its behavior, allowing you to improve the service's tagging over time.

The Saplo 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:



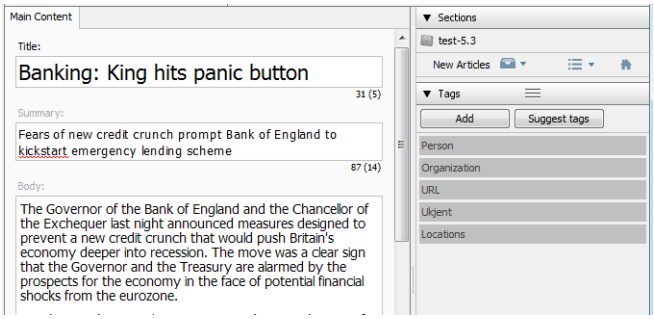
Clicking on the **Suggest tags** button submits the text of the current content item to Saplo, and adds the returned tags to the content item.

1.1 Prerequisites

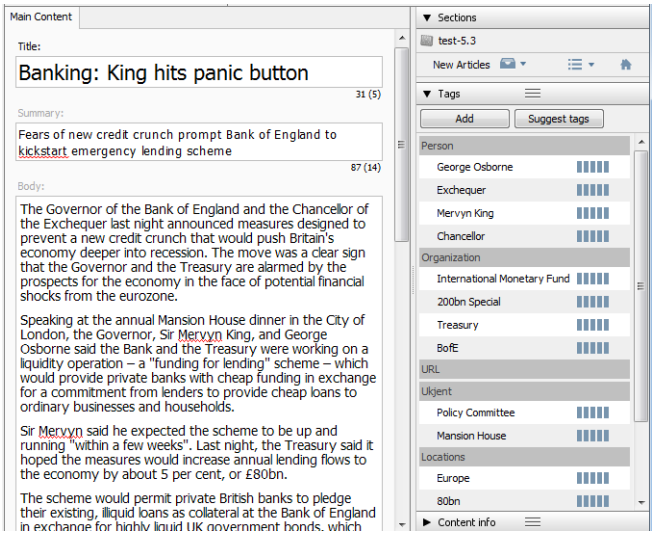
The only pre-requisite for using the Saplo Semantic plug-in is a Saplo account. In order for Saplo to work well with your publications you have to train it after setting up your account. Training involves submitting samples of your publication content to create a text **collection** that Saplo can analyse, and correcting the tags it generates from this collection. For further information see http://saplo.com/.

2 Using the Saplo Semantic Plug-in

Once the Saplo 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.



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



If Saplo 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 Saplo, 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 Saplo
- Change the relevance values set by Saplo

Any changes you make are sent back to Saplo, and will be used to improve its future suggestions.

2.1 Managing Tags

Saplo generates five different categories of tags:

- Persons
- Organizations
- Locations
- URLs
- Other

and the Saplo Semantic plug-in is configured to assign each of these categories to corresponding tag collections in your publication. If a tag suggestion returned by Saplo 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 Saplo 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 Saplo 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 Saplo Semantic 1.3.1.159245 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-saplo-1.3.1.159245.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 Saplo Semantic Installation

Installing Saplo Semantic involves the following steps:

1. Log in as **`escenic`** on your **assembly-host**.
2. Download the Saplo 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/5.7.0.148571/
saplo-semantic-1.3.1.159245.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 Saplo Semantic distribution file:

```
$ cd /opt/escenic/engine/plugins
$ unzip /mnt/download/saplo-semantic-1.3.1.159245.zip
```

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

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

3.3 Saplo Semantic Feedback Changelog Daemon Installation

This will install the feedback loop to Saplo. It is based on the changelog daemon framework that needs to be installed first.

Installing the Saplo Semantic feedback changelog daemon agent involves the following steps:

1. Unpack the changelog daemon distribution in an appropriate location
2. Start the daemon at least once with the following command

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

This will create a few directories that you will need, and create some files you need to edit

3. You will now need to edit the **`.daemon-conf/Daemon.properties`** file

There are properties you will need to change in this file:

`url`
This is the URI to the changelog for your installation. 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 the publication id of the publication you will provide feedback to Saplo from.

`username`
The username of the user that will be used to log into the webservice to get access to the changelog

`password`
The password for the above user

You will now have a configured changelog daemon ready to get an agent to run in it.

4. You will now need to copy the content of **`/opt/escenic/engine/plugin/semantic-saplo/misc/changlog/lib`** into into the empty **`daemon-install-dir/lib`** directory.
5. Issue the following command that will create directories

```
$ mkdir -p classes classes/com/escenic/daemon/
```

6. In order to use Saplo you must set up an account. The log-in credentials for this account must then be added to a configuration file so that the Saplo Semantic changelog daemon can use the account. To set up the required log-in configuration:
   1. Visit http://www.saplo.com and set up an account.
   2. Create a configuration file by copying **`/opt/escenic/engine/plugins/semantic-saplo/misc/example/SaploFeedbackAgent.properties`** to your newly created directory:

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

   3. Open your configuration file for editing.
   4. Uncomment the property assignments in the file and enter the API key, secret key and collection id supplied by Saplo:

```
apiKey=saplo-api-key
secretKey=saplo-secret-key
collectionId=collectionId
```

   5. Create a configuration file by copying **`/opt/escenic/engine/plugins/semantic-saplo/misc/example/CacheManager.properties`** to your newly created directory:

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

   6. Open your configuration file for editing.
   7. The schemes of the Content Engine tag structures that you want to be mapped to the Saplo categories:

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

```
category.unknown=tag:unknown@example.com,2011
category.url=tag:url@example.com,2011
```

The Content Engine tag structures you map on to Saplo 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.

8. Now stuff should just work. Start the daemon with

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

and watch it run. It might be a good idea to add it to the start-up script of your server.

This is a note about please read more about all this in the changelog daemon documentation..

3.4 Saplo Semantic Ontology Changelog Daemon Installation

This will install the ontology agent to Saplo. It is based on the changelog daemon framework that needs to be installed first.

Installing the Saplo Semantic ontology changelog daemon agent involves the following steps:

1. Unpack the changelog daemon distribution in an appropriate location
2. Start the daemon at least once with the following command

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

This will create a few directories that you will need, and create some files you need to edit

3. You will now need to edit the **.daemon-conf/Daemon.properties** file

There are properties you will need to change in this file:

url
This is the URI to the changelog for your installation. 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 the publication id of the publication you will provide feedback to Saplo from.

username
The username of the user that will be used to log into the webservice to get access to the changelog

password
The password for the above user

You will now have a configured changelog daemon ready to get an agent to run in it.

4. You will now need to copy the content of **/opt/escenic/engine/plugin/semantic-saplo/misc/changlog/lib** into into the empty **daemon-install-dir/lib direcotry**.
5. Issue the following command that will create directories

```
$ mkdir -p classes classes/com/escenic/daemon/
```

6. In order to use Saplo you must set up an account. The log-in credentials for this account must then be added to a configuration file so that the Saplo Semantic changelog daemon can use the account. To set up the required log-in configuration:
   1. Visit http://www.saplo.com and set up an account.
   2. Create a configuration file by copying **`/opt/escenic/engine/plugins/semantic-saplo/misc/example/SaploOntologyAgent.properties`** to your newly created directory:

      ```
      $ cp /opt/escenic/engine/plugins/semantic-saplo/misc/example/
      SaploOntologyAgent.properties \
            classes/com/escenic/daemon/
      ```
   3. Open your configuration file for editing.
   4. Uncomment the property assignments in the file and enter the API key, secret key and ontology id supplied by Saplo:

      ```
      apiKey=saplo-api-key
      secretKey=saplo-secret-key
      ontologyId=ontologyId
      ```
   5. Create a configuration file by copying **`/opt/escenic/engine/plugins/semantic-saplo/misc/example/SaploOntologyLoadService.properties`** to your newly created directory:

      ```
      $ cp /opt/escenic/engine/plugins/semantic-saplo/misc/example/
      SaploOntologyLoadService.properties \
            classes/com/escenic/daemon/
      ```
   6. Open your configuration file for editing.
   7. Uncomment the property assignment in the file and enter the interval in seconds to load ontology in Saplo.

      ```
      loadInterval=loadInterval
      ```
   8. Now stuff should just work. Start the daemon with

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

      and watch it run. It might be a good idea to add it to the start-up script of your server.

> This is a note about please read more about all this in the changelog daemon documentation..

3.5 Verify the Installation

To verify the status of the Saplo 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 Saplo Semantic after installing it (see chapter 3) you must:

1. Create a configuration for logging into the Saplo service
2. Configure Content Studio with mappings between your Content Engine tag collections and Saplo's tag categories.
3. Configure the content types that are to be tagged by Saplo

4.1 Configure Saplo Log-in Credentials

In order to use Saplo you must set up an account. The log-in credentials for this account must then be added to a configuration file so that the Saplo Semantic plug-in can use the account. To set up the required log-in configuration:

1. Visit http://www.saplo.com and set up an account.
2. Create a configuration file by copying **`/opt/escenic/engine/plugins/semantic-saplo/misc/siteconfig/com/escenic/webservice/proxy/SaploConfig.properties`** to your common configuration layer. For example:

```
$ cp /opt/escenic/engine/plugins/semantic-saplo/misc/siteconfig/com/escenic/
webservice/proxy/SaploConfig.properties \
  /etc/escenic/engine/common/com/escenic/webservice/proxy/SaploConfig.properties
```

3. Open your configuration file for editing.
4. Uncomment the two property assignments in the file and enter the API key and secret key supplied by Saplo:

```
replaceMapping.@@api_key@@ = saplo-api-key
replaceMapping.@@secret_key@@ = saplo-secret-key
```

4.2 Configure Content Studio

In order for Content Studio to be able to use the Saplo plug-in you must add the following settings to **`com/escenic/webstart/StudioConfig.properties`** in your common configuration layer:

`property.com.escenic.semantic.saplo.collection-id`
The ID of the Saplo collection you created when setting up your Saplo account. For example:

```
property.com.escenic.semantic.saplo.collection-id=1234
```

`property.com.escenic.semantic.saplo.ontology-ids`
The IDs of the Saplo ontology you created when setting up your Saplo account. For example:

```
property.com.escenic.semantic.saplo.ontology-ids=12,34
```

`property.com.escenic.semantic.saplo.category.organization`
The **scheme** of the Content Engine tag structure you want to be mapped to the Saplo "organizations" category. For example:

```
property.com.escenic.semantic.saplo.category.organization=tag:organization@example.com,2011
```

property.com.escenic.semantic.saplo.category.location
The **scheme** of the Content Engine tag structure you want to be mapped to the Saplo "locations" category. For example:

```
property.com.escenic.semantic.saplo.category.location=tag:location@example.com,2011
```

property.com.escenic.semantic.saplo.category.url
The **scheme** of the Content Engine tag structure you want to be mapped to the Saplo "URLs" category. For example:

```
property.com.escenic.semantic.saplo.category.url=tag:url@example.com,2011
```

property.com.escenic.semantic.saplo.category.other
The **scheme** of the Content Engine tag structure you want to be mapped to the Saplo "other" category. For example:

```
property.com.escenic.semantic.saplo.category.other=tag:other@example.com,2011
```

property.com.escenic.semantic.saplo.min-relevance
A minimum relevance below which tag suggestions will be rejected. Only tags which are assigned at least this relevance by Saplo will be accepted as suggestions by the Saplo Semantic plug-in. The specified value must be a number between 0 (no relevance) and 1 (maximum relevance). For example:

```
property.com.escenic.semantic.saplo.min-relevance=0.5
```

property.com.escenic.semantic.saplo.timeout
The maximum time (in seconds) that the Saplo Semantic plug-in will wait for suggestions from Saplo. If Saplo does not respond before the timeout expires then the auto-tagging operation is abandoned. For example:

```
property.com.escenic.semantic.saplo.timeout=60
```

The Content Engine tag structures you map on to Saplo 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.

4.3 Configure Content Types

The Saplo 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 Saplo

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 Saplo tag categories.

4.3.1 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-saplo`**. This is an empty element with one attribute, **`name`**. **`name`** may contain one of the following values:

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

`body`
: The contents of this field will be submitted for tagging in Saplo'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-saplo" name="body"/>
</field>
```

4.3.2 Adding a Response Field

A response field is required in each content type to hold responses returned from Saplo. 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="saplo-response">
  <ui:label>Saplo response</ui:label>
  <ui:description>The JSON response from saplo.</ui:description>
  <ui:hidden/>
  <semantic:field name="response"/>
</field>
```

4.3.3 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-saplo"
                xmlns:semantic="http://xmlns.escenic.com/2011/semantic"
                xmlns:saplo="http://xmlns.escenic.com/2011/semantic-saplo">
```

```
    <ui:label>Saplo Story</ui:label>
    <ui:description>A news story auto tagged by saplo</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>
        <saplo:field name="headline"/>
      </field>
      <field mime-type="text/plain" type="basic" name="summary">
        <ui:label>Summary</ui:label>
        <saplo:field name="body"/>
      </field>
      <field mime-type="application/xhtml+xml" type="basic" name="body">
        <ui:label>Body</ui:label>
        <saplo:field name="body"/>
      </field>
      <field mime-type="application/json" type="basic" name="saplo-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>
```