CUE Semantic

User Guide

1.9.0-5

Table of Contents

1 Introduction

CUE Semantic is a CUE extension that provides semi-automatic content tagging. If CUE Semantic is installed, then a **Semantic Analysis** section is added to the content editor's metadata panel, and used to display a list of suggested tags for the content currently displayed in the editor.

The list of suggestions is constantly updated to reflect changes in the content. Users can then tag content items by simply selecting suggestions from this list.

> CUE Semantic does not actually perform semantic analysis itself – rather, it offers seamless integration with an external semantic analysis service. In order to use CUE Semantic, therefore, you must separately purchase access to one of the semantic analysis services supported by CUE Semantic. The services currently supported by CUE Semantic are:
>
> - Atilika
> - DC Semantic Engine Services

CUE Semantic is based on a Java framework for building web services called Dropwizard.

1.1 How CUE Semantic Works

CUE Semantic works as follows:

- Content in the CUE editor is sent at regular intervals to the external service for analysis (every 10 seconds, by default).
- The external service returns a list of suggested tags for the submitted content, which are displayed in the **Semantic Analysis** section of the metadata panel.
- Information about tag selections made by users can be sent back to the external service. This information can then be used by the external service to improve future suggestions.

1.2 The Back End Services

The semantic analysis services supported by CUE Semantic are broadly similar but not identical. They provide slightly different features and use slightly different terminology. The following sections briefly describe the services they provide, and how they map onto the CUE Semantic interface.

CUE Semantic's interface is the same whichever back end service is used, and is more closely aligned with Atilika than DC, due to the fact that the Atilika interface was implemented first.

1.2.1 Atilika

The Atilika service returns two different types of tag suggestions:

Entities
Entities are tags referencing named entities in the real world, such as persons, places, organisations and so on. Atilika entities are defined by Atilika, and completely in Atilika's

control. You cannot change the entity-based tag suggestions made by Atilika. Atilika entities are grouped into the following categories:

- **PER** (persons such as Nelson Mandela, Donald Trump, etc.)
- **LOC** (locations such as Africa, Spain, New York, Chelsea)
- **ORG** (organizations such as companies, charities, political parties and so on)
- **TTL** (titles such as Mr, Dr, Sir etc.)
- **FAC** (facilities such as hotels, sports grounds, theatres, universities, shopping malls)
- **REL** (religions)
- **NAT** (nationalities such as Irish, Indian, Australian)

Concepts

Concepts are more abstract tags, referencing ideas or topics such as politics, medicine, football, crime and so on. Concepts are not defined by Atilika, they are customer-defined tags. Atilika learns how to apply concepts by example. The process can be started by supplying a large number of ready-tagged documents to Atilika. Atilika learns from this sample what concepts to suggest for future content. If continuously supplied with feedback on how concept tags are used, it will continue learning and improving its suggestions.

CUE Semantic can be configured to make use of entities only, concepts only, or both entities and concepts.

1.2.2 DC Semantic Engine Services

The DC service returns two different types of tag suggestions. Both types of tag suggestion are defined by DC. You cannot change any of the tag suggestions made by DC.

Entities

Entities are tags referencing named entities in the real world, such as persons, places, organisations and so on. DC entities are grouped into the following categories:

entityMapping: - from: city to: LOC defaultRelevance: 0.6 - from: country to: LOC defaultRelevance: 0.4 - from: person to: PER defaultRelevance: 0.6 - from: organisation to: ORG defaultRelevance: 0.6 - from: category to: CON

- **person** (Nelson Mandela, Donald Trump, etc.) These entities are mapped to **PER**.
- **city** (New York, Paris, etc.) These entities are mapped to **LOC**.
- **country** (Russia, Uganda, etc.) These entities are mapped to **LOC**.
- **organisation** (companies, charities, political parties, etc.) These entities are mapped to **ORG**.

Categories

Categories are more abstract tags, referencing ideas or topics such as politics, medicine, football, crime and so on. Categories are mapped to the identifier **CON** (the same identifier as is used for Atilika concepts).

2 Installation

You can install CUE Semantic on any Ubuntu or RedHat / CentOS machine in your network. It does not need to be installed on the same machine as CUE (although it can be). Before you try to install it, make sure that:

- You have installed Java 11 or higher on the machine where CUE Semantic is to be installed.
- You have credentials for accessing Stibo DX's software repositories.
- You have credentials for accessing the Atilika semantic analysis service.

2.1 Installation Procedure

To install CUE Semantic on an Ubuntu system, do the following:

1. Log in as **`root`**.
2. If necessary, add the CUE **`apt`** repository to your list of sources:

   ```
   # echo "deb http://user:password@apt.escenic.com stable main non-free" >> /etc/
   apt/sources.list.d/escenic.list
   ```

   where *user* and *password* are your CUE download credentials (the same ones you use to access the CUE Maven repository). If you do not have any download credentials, please contact CUE support.
3. Enter the following commands:

   ```
   # apt-get update
   # apt-get install cue-semantic-enrichment-service
   ```
4. If you intend to use the feedback service to return training data to the semantic analysis service, then you also need to enter:

   ```
   # apt-get install cue-semantic-feedback-service
   ```

On RedHat / CentOS systems, enter the following command as **`root`**:

```
# rpm -Uvh https://user:password:yum.escenic.com/rpm/cue-semantic-enrichment-
service-1.9.0-5.x86_64.rpm
```

to install the main service, and **if required**:

```
# rpm -Uvh https://user:password:yum.escenic.com/rpm/cue-semantic-feedback-
service-1.9.0-5.x86_64.rpm
```

to install the feedback service.

> Note that:
>
> - The feedback service can only be used together with an Atilika back end.
> - Even if you intend to use an Atilika back end, use of the feedback service is optional.

2.2 Configuration

After installation, you need to configure both CUE Semantic itself and CUE.

2.2.1 Configuring CUE Semantic

CUE Semantic consists of two components:

`cue-semantic-enrichment-service`
This service is responsible for getting tag suggestions from the external service and must be configured.

`cue-semantic-feedback-service`
This optional service is responsible for sending feedback to the external service. It only needs to be configured if you have installed it and intend to use it.

Before starting, make sure you have obtained the credentials you need to access Atilika You will need a username and password, plus URLs for one or more of the following service endpoints, depending on which Atilika features you intend to make use of:

- Entity tag suggestions
- Concept tag suggestions
- The feedback service

2.2.1.1 Configuring the Enrichment Service (Atilika)

To configure **`cue-semantic-enrichment-service`** for use with an Atilika back end:

1. Log in as **`root`**.
2. Open **`/etc/escenic/cue-semantic-enrichment-service/cue-semantic-enrichment-service.yaml`** in an editor.
3. Replace the **`providers`** group of settings with the following:

```
providers:
  - type: atilika_entities
    url: atilika-suggest-entities-url
    relevanceThreshold: 0.3
    connectTimeout: 5000
    requestTimeout: 5000
    credentials:
      username: atilika-username
      password: atilika-password
  - type: atilika_concepts
    url: atilika-suggest-concepts-url
    relevanceThreshold: 0.3
    connectTimeout: 5000
    requestTimeout: 5000
    credentials:
      username: atilika-username
```

```
      password: atilika-password
```

where *atilika-suggest-entities-url*, *atilika-suggest-concepts-url, atilika-username* and *atilika-password* are access credentials supplied by Atilika. The default settings shown above should work in most circumstances for the other properties:

`relevanceThreshold`
This property controls how many suggestions are returned from Atilika. Atilika gives each entity and concept it returns a relevance score between 0 (not relevant) and 1 (very relevant). A **`relevanceThreshold`** setting of 0.3 rejects all suggestions that do not have a relevance score of 0.3 or higher.

`connectTimeout`
The interval after which a connection attempt is abandoned, specified in milliseconds.

`connectTimeout`
The interval after which a request for tag suggestions is abandoned, specified in milliseconds.

If you do not intend to make use of either entities or concepts, then you can omit the corresponding group of settings. If, for example you only intend to use entities, then you would only need to enter:

```
providers:
  - type: atilika_entities
    url: atilika-suggest-entities-url
    relevanceThreshold: 0.3
    connectTimeout: 5000
    requestTimeout: 5000
    credentials:
      username: atilika-username
      password: atilika-password
```

4. Edit the following entries as required:

```
fields-to-analyze:
  - title
  - body

story-elements-to-analyze:
  - lead_text
  - paragraph
  - pull_quote
  - headline
```

`fields-to-analyze` determines which fields in classic rich text-based content items are sent to the external service for analysis. **`story-elements-to-analyze`** determines which story elements from storyline-based content items are sent to the external service for analysis.

5. By default, **`cue-semantic-enrichment-service`** runs on ports **`9180`** and **`9181`**. If you need to, you can change this by modifying the following properties:

```
server:
  applicationConnectors:
    - type: http
      port: 9180

  adminConnectors:
    - type : http
      port : 9181
```

6. Save your changes.

2.2.1.1.1 Concept Field Mappings

When analyzing text for concepts, Atilika distinguishes between body text, titles, teasers and so on. You can therefore improve its performance by assigning the CUE fields and story elements to be analyzed to the correct Atilika field types. To do this, add a **`mappings`** entry in the **`type: atilika_concepts`** section of the configuration file, as follows:

```
providers:
  - type : atilika_entities
    ...
  - type : atilika_concepts
    url : ...
    ...
    mappings:
      title:
       - title
       - headline
      teaser:
       - lead_text
```

Under **`mappings`**, the entries must be the names of the field types defined by Atilika (see the Atilika documentation for a complete up-to-date list of supported types). Under each of these field type entries, list the names of the fields and story elements you want to assign to that field type. Note that fields/story elements you specify here must also be specified under **`fields-to-analyze`** or **`story-elements-to-analyze`**, otherwise they won't be processed at all.

Any fields/story elements that are specified under **`fields-to-analyze`** or **`story-elements-to-analyze`** but not listed here will by default be assigned to the Atilika **`body`** field type. In the example shown above, the **`title`** field and **`headline`** story element will be assigned to the **`title`** Atilika field type, and the **`lead_text`** story element will be assigned to the **`teaser`** Atilika field type. All the other fields and story elements listed under **`fields-to-analyze`** and **`story-elements-to-analyze`** will be assigned to the **`body`** Atilika field type.

2.2.1.2 Configuring the Enrichment Service (DC Semantic Engine Services)

To configure **`cue-semantic-enrichment-service`** for use with a DC back end:

1. Log in as **`root`**.
2. Open **`/etc/escenic/cue-semantic-enrichment-service/cue-semantic-enrichment-service.yaml`** in an editor.
3. Replace the **`providers`** group of settings with the following:

```
providers:
- type : dc
  url : dc-url
  relevanceThreshold: 0.3
  credentials:
    apikey : dc-api-key
  entityMapping:
   - from: city
     to: LOC
     defaultRelevance: 0.6
   - from: country
     to: LOC
```

```
      defaultRelevance: 0.4
    - from: person
      to: PER
      defaultRelevance: 0.6
    - from: organisation
      to: ORG
      defaultRelevance: 0.6
    - from: category
      to: CON
```

where *dc-url* and *dc-api-key* are the access credentials supplied by DC. The default settings shown above should work in most circumstances for the other properties:

`relevanceThreshold`
This property controls how many category suggestions are returned from DC. DC gives each category (**`CON`**) suggestion it returns a relevance score between 0 (not relevant) and 1 (very relevant). A **`relevanceThreshold`** setting of 0.3 rejects all suggestions that do not have a relevance score of 0.3 or higher.

`entityMapping`
This section defines the mapping between the DC entity/category types and CUE Semantic's suggestion ids. You should not change these. Note, however, that the **`city`**, **`country`**, **`person`** and **`organisation`** entries each include a **`defaultRelevance`** property. This property is needed because DC does not return a relevance score that can be used by CUE Semantic with its entities. The specified default relevance is used instead. You must specify a **`defaultRelevance`** between **`0`** and **`1`**. If you do not specify a **`defaultRelevance`** that has a value between **`0`** and **`1`**, then the entities will not be used. The **`category`** entry does not need a **`defaultRelevance`** setting, because DC returns a relevance score with categories.

4. Edit the following entries as required:

```
fields-to-analyze:
  - title
  - body

story-elements-to-analyze:
  - lead_text
  - paragraph
  - pull_quote
  - headline
```

`fields-to-analyze` determines which fields in classic rich text-based content items are sent to the external service for analysis. **`story-elements-to-analyze`** determines which story elements from storyline-based content items are sent to the external service for analysis.

5. By default, **`cue-semantic-enrichment-service`** runs on ports **`9180`** and **`9181`**. If you need to, you can change this by modifying the following properties:

```
server:
  applicationConnectors:
    - type: http
      port: 9180

  adminConnectors:
    - type : http
      port : 9181
```

6. Save your changes.

2.2.1.3 Configuring the Feedback Service (Atilika only)

You only need to configure this service if:

- You intend to make use of Atilika concepts
- You have installed the feedback service

The feedback service consists of two components:

- A **daemon** that monitors a publication for new events. This daemon is implemented using another Stibo DX product called the Change Log Daemon. For full information about this product and its capabilities, see Change Log Daemon.
- An **agent** that selects concept tag creation events from the data collected by the daemon and returns information about them to to the external service. Note that only information about concept tags is returned to the external service.

To configure **`cue-semantic-feedback-service`**:

1. Log in as **`root`**.
2. Open **`/etc/escenic/cue-semantic-feedback-service/Daemon.properties`** in an editor.
3. Add the following entries to the file:

```
urls = changelog-url
username = cue-username
password = cue-password
```

   where:

   - *changelog-url* is the URL of a publication change log. A publication change log URL looks like this:

```
http://content-store-host/webservice/escenic/changelog/
publication/publicationId
```

   - *username* is a user name to be used for logging into the Content Store and accessing the change log.
   - *password* is the password for the specified user name.

   The Change Log Daemon is actually capable of monitoring the change logs of more than one publication. If you need to be able to do that, see Complex Configurations to find out how.

4. Save your changes.
5. Create a file called **`/etc/escenic/cue-semantic-feedback-service/Agent.properties`** and open it in an editor.
6. Add the following entries to the file:

```
$class=com.escenic.semantic.atilika.FeedbackAgent
fieldsToAnalyze = title,body
storyElementsToAnalyze = lead_text,paragraph,pull_quote, headline
url= atilika-feedback-concepts-url
username = atilika-username
password = atilika-password
```

```
conceptsTagScheme = tag:topic@escenic.com,2017
```

Note that:

- The lists in **`fieldsToAnalyze`** and **`storyElementsToAnalyze`** must match the corresponding fields in **`/etc/escenic/cue-semantic-enrichment-service/cue-semantic-enrichment-service.yaml`** (see section 2.2.1).
- *atilika-feedback-concepts-url*, *atilika-username* and *atilika-password* must be access credentials supplied by Atilika.
- The tag scheme specified for **`conceptsTagScheme`** must be the one to which Atilika **`CON`** suggestions are mapped in **`/etc/escenic/cue-web/config.yml`** (see section 2.2.2).

7. Optionally, make the following additional entries:

```
mapping.title=title,headline
mapping.teaser=lead_text
```

These entries are only required if you are making use of Atilika concepts, and should match the corresponding mapping entries added to the enrichment service configuration file (see section 2.2.1.1.1).

8. Save your changes.

2.2.2 Configuring CUE

To make CUE use CUE Semantic:

1. Log in as **`root`**.
2. Copy the example file included with the distribution of CUE located in **`/etc/escenic/cue-web/public/Semantic.yml`** and uncomment all the lines. The example file has this content:

```
enrichmentServices:
  - href: "http://semantic-service-host/"
    name: "Extract Semantic Entities"
    title: "Extract Semantic Entities"
    triggers:
      - name: editor-recurring
        properties:
          interval: 10
          timeout: 20
editors:
  metadata:
    - name: 'cue-semantic-metadata-panel'
      directive: 'cue-semantic-metadata-panel'
      cssClass: 'semantic'
      title: 'Semantic Analysis' #translate
      mimeTypes: ['x-ece/story', 'x-ece/new-content; type=story']
      args: "[content]='ngModel.resource.data' [content-
values]='ngModel.resource.data.values' [lock-
helper]='ngModel.lockHelper' [locks]='ngModel.lockHelper.locks'"
      requires: ['escenic']
      order: 850
      isAngular: true
      capability: 'cue.metadata.semantic'
semanticEntityTypes:
  - name: PER
    tagScheme: tag:person@escenic.com,2017
    title: Persons
```

```
  - name: LOC
    tagScheme: tag:location@escenic.com,2017
    title: Locations
  - name: ORG
    tagScheme: tag:organization@escenic.com,2017
    title: Organizations
  - name: TTL
    tagScheme: tag:title@escenic.com,2017
    title: Titles
  - name: FAC
    tagScheme: tag:facility@escenic.com,2017
    title: Facilities
  - name: REL
    tagScheme: tag:religion@escenic.com,2017
    title: Religion
  - name: NAT
    tagScheme: tag:nationality@escenic.com,2017
    title: Nationalities
  - name: CON
    tagScheme: tag:concept@escenic.com,2017
    title: Concepts
```

Replace *semantic-service-host* with the host name or IP address of the host on which you installed CUE Semantic.

3. You may then want to modify the entries highlighted in **bold**:

   `interval`
   This property determines how frequently CUE Semantic requests new tag suggestions from the external service, specified in seconds.

   `timeout`
   This property determines how long CUE Semantic waits for a response from the external service before a "can't connect" message is displayed in the CUE Notification Center.

   `semanticEntityTypes`
   This entry contains an array of **`name`** / **`tagScheme`** / **`title`** properties that map the suggestions returned by the external service to Content Store tag schemes. Atilika **concepts**/DC **categories** are returned under the name **`CON`**. Atilika/DC **entities** are returned under the following names:

   - **`PER`** (persons such as Nelson Mandela, Donald Trump, etc.)
   - **`LOC`** (locations such as Africa, Spain, New York, Chelsea)
   - **`ORG`** (organizations such as companies, charities, political parties and so on)
   - **`TTL`** (titles such as Mr, Dr, Sir etc. – from Atilika only)
   - **`FAC`** (facilities such as hotels, sports grounds, theatres, universities, shopping malls – from Atilika only)
   - **`REL`** (religions – from Atilika only)
   - **`NAT`** (nationalities such as Irish, Indian, Australian – from Atilika only)

   You can map each of these categories onto a separate tag scheme, or map multiple categories onto the same tag scheme if you wish, as is the case for **`TTL`**, **`FAC`**, **`REL`** and **`NAT`** in the example shown above. You can also choose to ignore a category by omitting a mapping for it. If, for example, you remove:

   ```
   - name: TTL
   ```

```
tagScheme: tag:entity@escenic.com,2017
title: Titles
```

from the example given above, then the **TTL** category will be ignored and no **title** tag suggestions will appear in CUE.

> When using an Atilika back end, concepts (**CON**) (if used) should always be mapped to their own tag scheme that is not used for any other purpose. This is because tags in this tag scheme are the ones returned to Atilika as feedback.

The tag schemes you reference in this section must already exist in the Content Store. For information on how to create tag schemes, see Manage Tag Structures.

4. Save your changes.

You will also probably want to change where the **Semantic Analysis** section appears in the metadata panel. You can control what sections appear in the metadata panel and the order in which they appear on a per-content type basis. For information on how to do this, see Metadata Panel Sections. The name to use for the **Semantic Analysis** section is **cue-semantic-metadata-panel**, as defined in **/etc/escenic/cue-web/public/Semantic.yml** (above).

2.3 Running CUE Semantic

Once it has been installed and configured, CUE Semantic can be started as a service with the following **systemd** commands:

```
systemctl start cue-semantic-enrichment-service
systemctl enable cue-semantic-enrichment-service
systemctl start cue-semantic-feedback
systemctl enable cue-semantic-feedback
```

2.4 Troubleshooting

Both **cue-semantic-enrichment-service** and **cue-semantic-feedback-service** service maintain error logs:

/var/log/escenic/cue-semantic-enrichment-service.log
/var/log/escenic/cue-semantic-feedback-service.log

If one of the services is not running properly, check the appropriate log file for hints.

For the enrichment service, you can control logging levels by changing the settings in **/etc/escenic/cue-semantic-enrichment-service/cue-semantic-enrichment-service.yaml**:

```
logging:
  level: INFO
  loggers:
    io.dropwizard: INFO
    com.escenic.semantic : INFO
  appenders:
    - type: file
      currentLogFilename: /var/log/escenic/cue-semantic-enrichment-service.log
```

```
      archivedLogFilenamePattern: /var/log/escenic/cue-semantic-enrichment-service-
%d.log.gz
      archivedFileCount: 5
      timeZone: UTC
```

For the feedback service, you can configure extra logging by adding a **$log** property to **/etc/escenic/cue-semantic-feedback-service/Agent.properties**:

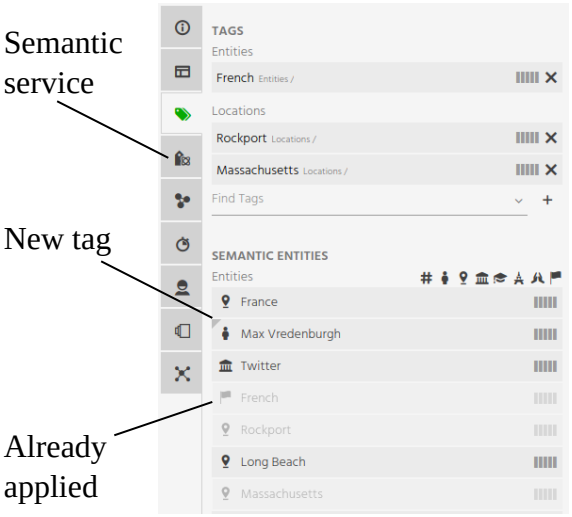
```
$log = INFO
```

The feedback service also maintains a state file, **/var/lib/escenic/cue-semantic-feedback-service.state.properties**. This state file stores the current position in the change log. If you want to restart from the beginning, remove this file.

> In the case where a service fails completely and doesn't even start, no log file will be generated. In this case, check the system log **/var/log/syslog**. This can happen, for example, if there is a typo in the enrichment service configuration file, **cue-semantic-enrichment-service.yaml**.

3 Using CUE Semantic

If CUE Semantic is installed and running, then a **Semantic Analysis** section is added to the CUE content editor's metadata panel, and used to display a list of suggested tags for the content currently displayed in the editor:



The list of suggestions is constantly updated to reflect changes in the content. To tag a content item, all you have to do is select suggestions from this list by clicking on them. When you select a suggestion, it is grayed out to indicate that it is now selected, and it immediately appears in the **Tags** section of the metadata panel as a tag. Once it is added here as a tag, you can optionally adjust the relevance score suggested by the external service.

When a suggestion is accepted, one of two things can happen:

- If the suggestion **exactly** matches the label of an existing tag in the tag schema to which the suggestion has been mapped, then that tag is used to tag the content item.
- If the suggestion does not exactly match the label of an existing tag in the tag schema to which the suggestion has been mapped, then a new tag is added to the tag schema.

Note that:

- The tag label matching is case sensitive. If your tag schema contains a tag called **football**, accepting a suggestion with the name **Football** will result in the creation of a new tag.
- New tags are always added at the root level of a tag schema.

For these reasons, it may be necessary to periodically tidy up your tag schemas by merging tags and moving them from the root level to the appropriate location in tree-structured tag schemas, especially if you use CUE Semantic to update existing tag schemas that contain tags not suggested by CUE Semantic's external service.

CUE Semantic tries to encourage consistent tagging by:

- Marking suggested tags that do not already exist in the tag schema as new (they are marked with a triangular tab as shown in the screenshot above)

- Asking for confirmation when you add a new tag.

Before adding a new tag, you should always check to see if there is another semantically equivalent tag that already exists in the tag schema. In this way you will avoid the need for merging tags later.