Fluenta

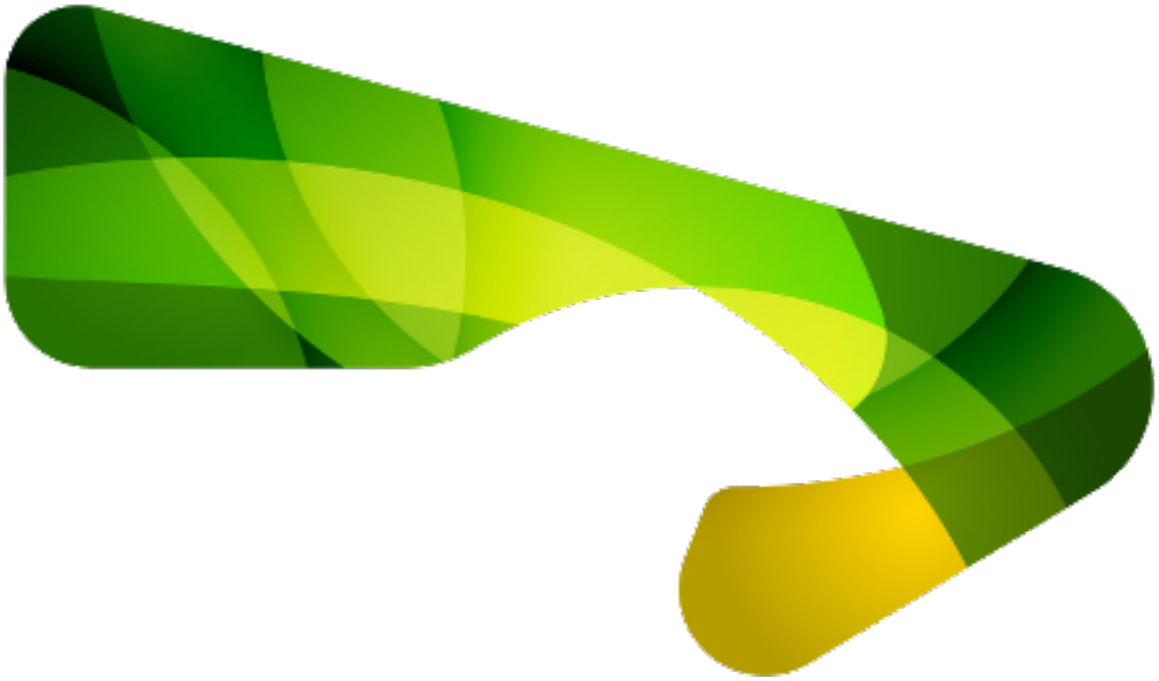


Table of Contents

Introduction

Fluenta is a tool designed to simplify the translation of DITA projects.

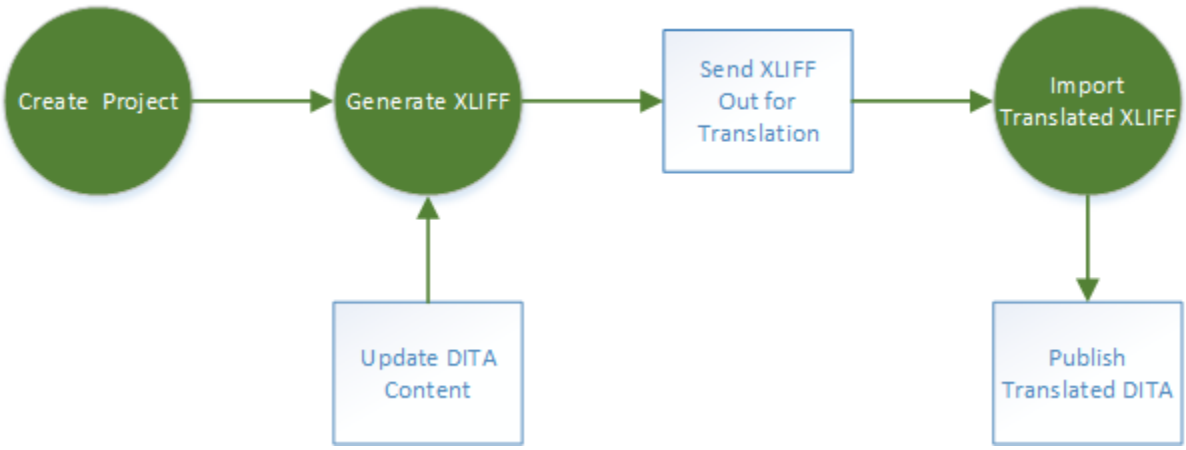
Fluenta is able to parse a DITA map, resolving the references to all topics and subtopics, preparing a unified XLIFF file that you can send to your Language Sevice Provider (LSP).

Fluenta implements the procedure for translating DITA projects recommentded by the OASIS DITA Adoption TC.

How it works

1. Start by creating a project. All you have to do is provide the location of your DITA map and select the languages that you want to translate into.
2. When you are ready to translate your project, generate an XLIFF file from it.
3. Send the XLIFF file to your Language Service Provider and wait for a translated XLIFF.
4. Import the translated XLIFF and select a folder where to store the translated version of your map and topics.

The four steps described above are all you need to get a translated version of your DITA project. The following diagram shows the processes involved:



After updating your DITA content, you may want to update the translations of your project. All you have to do at this moment is:

1. Generate a new XLIFF file.
2. Send the new XLIFF file to your Language Service Provider and wait for a translated XLIFF.
3. Import the translated XLIFF and select the folder where to store the updated translated version of your map and topics.

Fluenta automatically recovers In-Context Exact (ICE) matches from the translation that was stored in the previous cycle. This means that you don't need to pay again for the translation of content that didn't change.

Thanks to the Translation Memory technology included in Fluenta, you can also recover translations of parts that were slightly changed.

Translating DITA Projects

Translating DITA projects is a process that comprises these steps:

1. Create a project
2. Generate XLIFF files
3. Translate XLIFF files
4. Import translated XLIFF files

Create Project

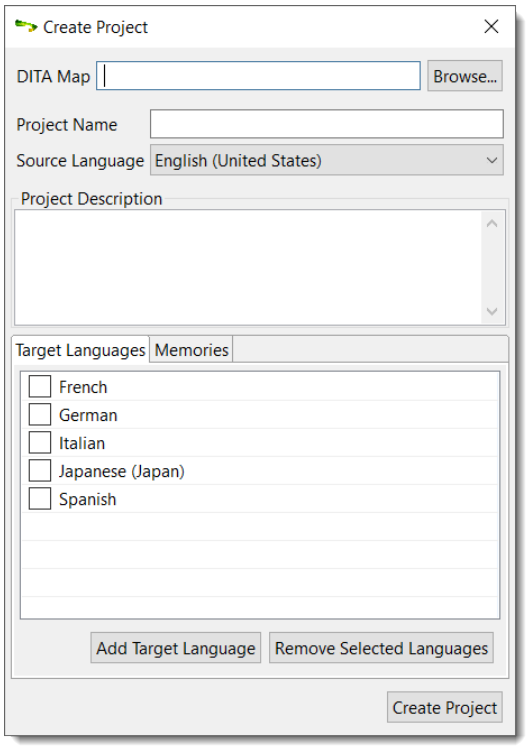
About this task

Follow these steps to create a translation project from your DITA map.

Procedure

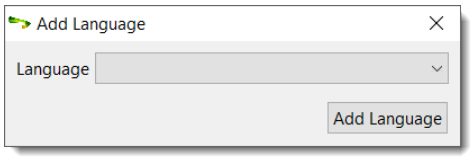
1. In **Projects** menu, select **Create Project** or click the **Create Project** button in **Projects** view toolbar.

   The following dialog appears:



2. Type the name of the DITA map in the **DITA Map** text box or use the **Browse...** button to select a DITA map from the file system.
3. Type a project name in the **Project Name** text box.
4. Optionally, enter a description for the project in the **Project Description** text box.
5. Add additional target languages to the default set if required.
   a. Click **Add Target Language** button.

The following dialog appears:



b. Select a language from the **Language** drop´down list.

c. Click the **Add Language** button.

Selected language is added to the project and the dialog is closed.

6. Remove unnecessary languages from the default language set if required.

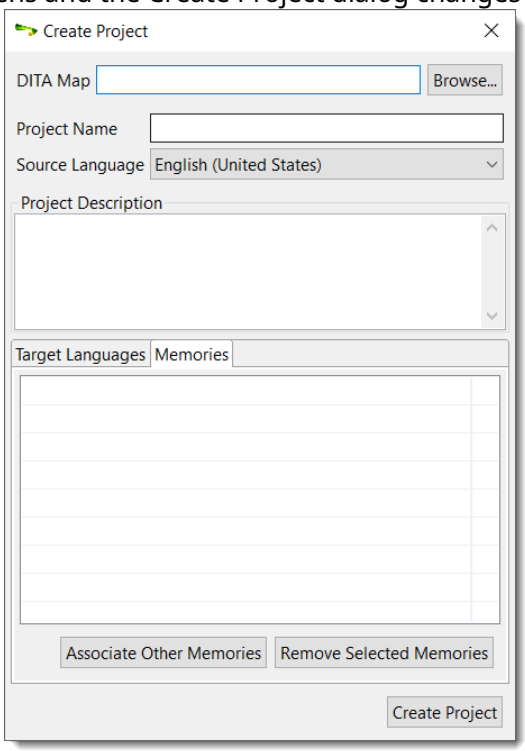
a. Select the check boxes next to each language you want to remove.

b. Click the **Remove Selected Languages** button.

7. Associate additional memories with the new project if required.

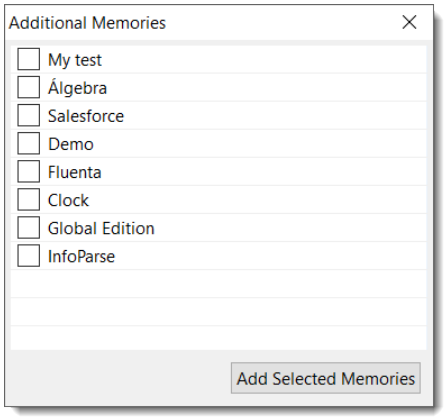
a. Click on the **Memories** tab.

The **Memories** tab opens and the Create Project dialog changes to:



b. Click the **Associate Other Memories** button.

The following dialog appears:



c. Select the check boxes next to the memories that you want to associate with the new project.

d. Click the **Add Selected Memories** button.

   Selected memories are associated with the new project and the dialog closes.

8. Click the **Create Project** button.

Results

A new project is created and the list of projects in the **Projects** view is updated to reflect the changes.

Generate XLIFF

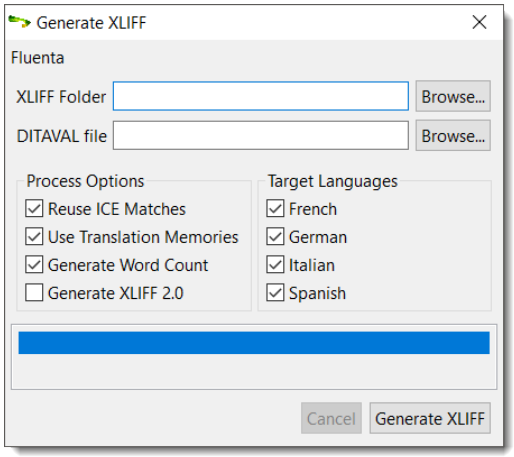
About this task

Follow these steps to generate XLIFF files that you can send to your Language Service Provider (LSP) for translating your projects.

Procedure

1. In **Projects** view, select the project that will be translated.
2. In **Projects** menu, select **Generate XLIFF** or click the **Generate XLIFF** button in the toolbar.

   The following dialog appears:



3. Type the name of the folder where the XLIFF files should be placed in the **XLIFF Folder** text box or use the **Browse...** button to select a folder in the file system.
4. If you need to exclude some topics, enter the name of a DITAVAL file that you want to use for conditional processing in the **DITAVAL File** text box or use the **Browse...** button next to it to select a file from the file system.
5. Select the **Reuse ICE Matches** check box if you want to compare current content with the content available last time an XLIFF file was generated and reuse all existing translations.
6. Select the **Use Translation Memories** check box if you want to recover translations for untranslated segments using the memories associated with the project.
7. Select the **Generate Word Count** check box if you want a statistic analysis to be generated for each target language.
8. Select the **Generate XLIFF 2.0** check box if you need to generate XLIFF 2.0 instead of XLIFF 1.2 (default).
9. Select the check boxes corresponding to the target languages that you want to process.
10. Click the **Generate XLIFF** button.

    Generation status is displayed in the progress panel and the **Cancel** button is enabled.

Results

An XLIFF file is generated for each selected target language. XLIFF files and optional word counts are placed in the selected XLIFF Folder.

Import XLIFF

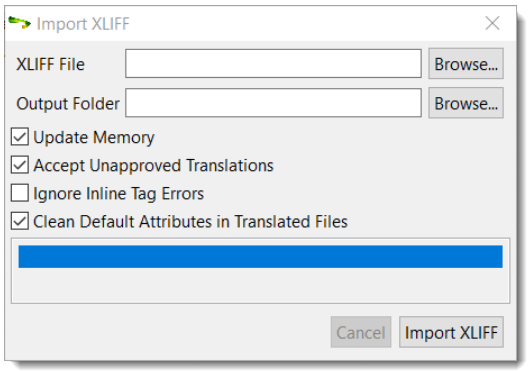
About this task

Once you receive a translated XLIFF from your Language Service Provider, you need to import it into Fluenta to generate a translated version of your project.

Procedure

1. In **Projects** view, select the project that will receive the translated data.
2. In **Projects** menu, select **Import XLIFF** or click the **Import XLIFF** button in the toolbar.

   The following dialog appears:



3. Type the name of the XLIFF file to import in the **XLIFF File** text box or use the **Browse...** button next to it to select an XLIFF file from the file system.

4. Type the name of the folder where the translated content should be placed in the **Output Folder** text box or use the **Browse...** button to select a folder in the file system.
5. Select the **Update Memory** check box if you want to store the imported translations in the project memory.
6. Select the **Accept Unapproved Translations** check box if the XLIFF file being imported does not have all translations marked as approved and you want to accept the existing drafts.
7. Select the **Ignore Inline Tag Errors** check box if you want to try importing an XLIFF file despite its errors with inline tags (may break the DITA content and make publication in target language impossible). If the check box is left blank and errors are found, a detailed report of errors in HTML format is automatically generated and displayed in the default browser.
8. Select the **Clean DefaultAttributes in Translated Files** check box if you want Fluenta to remove the attributes that are automatically added by XML parsers when validating against a DTD or XML Schema. Removed attributes are: `@class`, `@xmlns:ditaarch`, `@ditaarch:DITAArchVersion`, and `@domains`.
9. Click the **Import XLIFF** button.

   Import status is displayed in the progress panel and the **Cancel** button is enabled.

Results

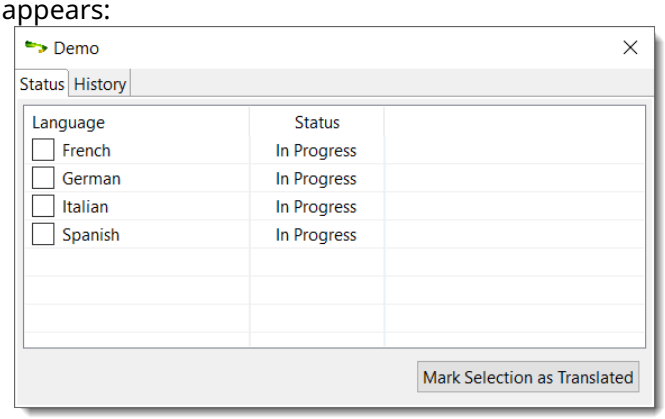
The XLIFF file is imported into the selected project. A translated version of the project content is created in the indicated output folder.

Project Status

Follow these steps to check the translation status of your projects.

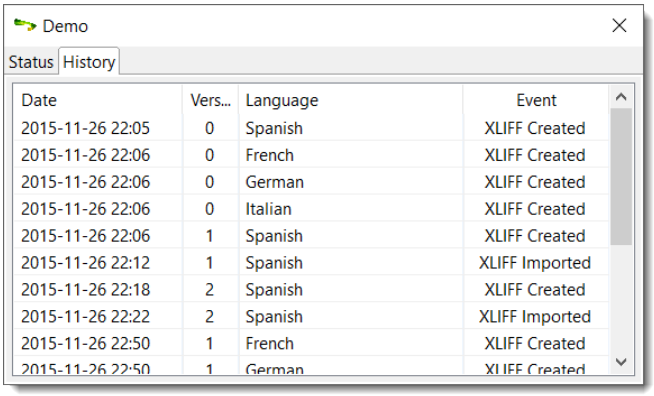
1. In **Projects** view, select the project that you want to examine.
2. In **Projects** menu, select **Project Information** or click the **Project Information** button in the toolbar.

The following dialog appears:



You can mark one or more languages as translated by selecting the corresponding check boxes and clicking the **Mark Selection as Translated** button.

Click the **History** tab to display a list of events associated with your project.


Demo
Status
History
Date
Vers...
Language
Event
2015-11-26 22:05
0
Spanish
XLIFF Created
2015-11-26 22:06
0
French
XLIFF Created
2015-11-26 22:06
0
German
XLIFF Created
2015-11-26 22:06
0
Italian
XLIFF Created
2015-11-26 22:06
1
Spanish
XLIFF Created
2015-11-26 22:12
1
Spanish
XLIFF Imported
2015-11-26 22:18
2
Spanish
XLIFF Created
2015-11-26 22:22
2
Spanish
XLIFF Imported
2015-11-26 22:50
1
French
XLIFF Created
2015-11-26 22:50
1
German
XLIFF Created

Translation Memories

Translation Memory (TM) is a language technology that enables the translation of segments (paragraphs, sentences or phrases) of documents by searching for similar segments in a database and suggesting matches that are found in the databases as possible translations.

When you create a project, a new translation memory is automatically created and associated with your project. When you import a translated XLIFF file, the memory associated with the project is populated with the translations included in the XLIFF file.

When you generate a new XLIFF file after adding new content to your project, Fluenta can reuse the data stored in the associated memories to translate the newly added content, reducing translation costs.

Create Memory

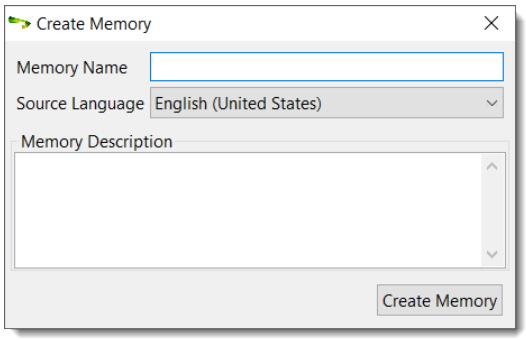
About this task

Follow these steps to create a new translation memory.

Procedure

1. In **Memories** menu, select **Create Memory** or click the **Create Memory** button in **Memories** view toolbar.

   The following dialog appears:



2. Type a name for the new memory in the **Memory Name** text box.
3. Select the source language for the new memory in the **Source Language** drop´down list.
4. Optionally, enter a description in the **Memory Description** text box.
5. Click the **Create Memory** button.

Results

A new memory is created and the list of memories in the **Memories** view is updated to reflect the changes.

Edit Memory

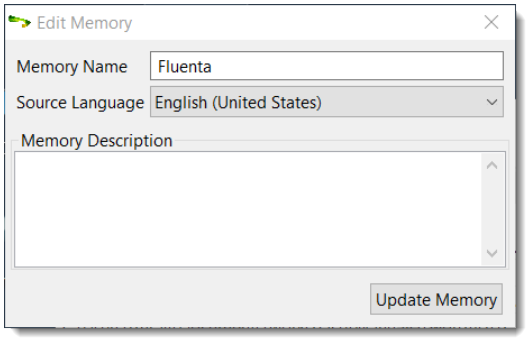
About this task

Follow these steps to edit the name, source language or description of a translation memory.

Procedure

1. In **Memories** menu, select **Edit Memory** or click the **Edit Memory** button in **Memories** view toolbar.

   The following dialog appears:



2. Edit all fields as required.
3. Click the **Update Memory** button.

Results

Selected memory data is updated and the list of memories in the **Memories** view is updated to reflect the changes.

Import Memory

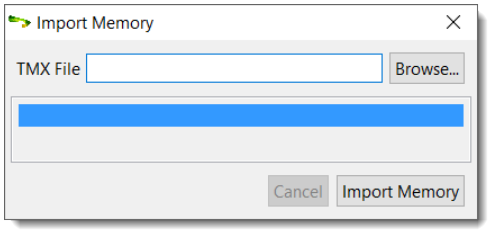
About this task

You can populate translation memories with data from TMX files. Use this feature if you have legacy TM data that you want to reuse in Fluenta.

Procedure

1. In **Memories** view, select the translation memory that will receive the imported data.
2. In **Memories** menu, select **Import TMX File** or click the **Import TMX File** button in the toolbar.

   The following dialog appears:



3. Type the name of the TMX file to import in the **TMX File** text box or use the **Browse...** button next to it to select a TMX file from the file system.
4. Click the **Import Memory** button.

   Import status is displayed in the progress panel and the **Cancel** button is enabled.

Results

The TMX file is imported into the selected translation memory.

Export Memory

About this task

The content of Fluenta memories can be exported in TMX format for exchanging with other tools or for backup purposes.

Procedure

1. In **Memories** view, select the translation memory that you want to export.
2. In **Memories** menu, select **Export TMX File** or click the **Export TMX File** button in the toolbar.

   A file selection dialog appears.
3. Select a name and location for the TMX file that will contain the memory data.
4. Click the **Save** button.

Results

Memory data is exported in TMX format in the selected file.

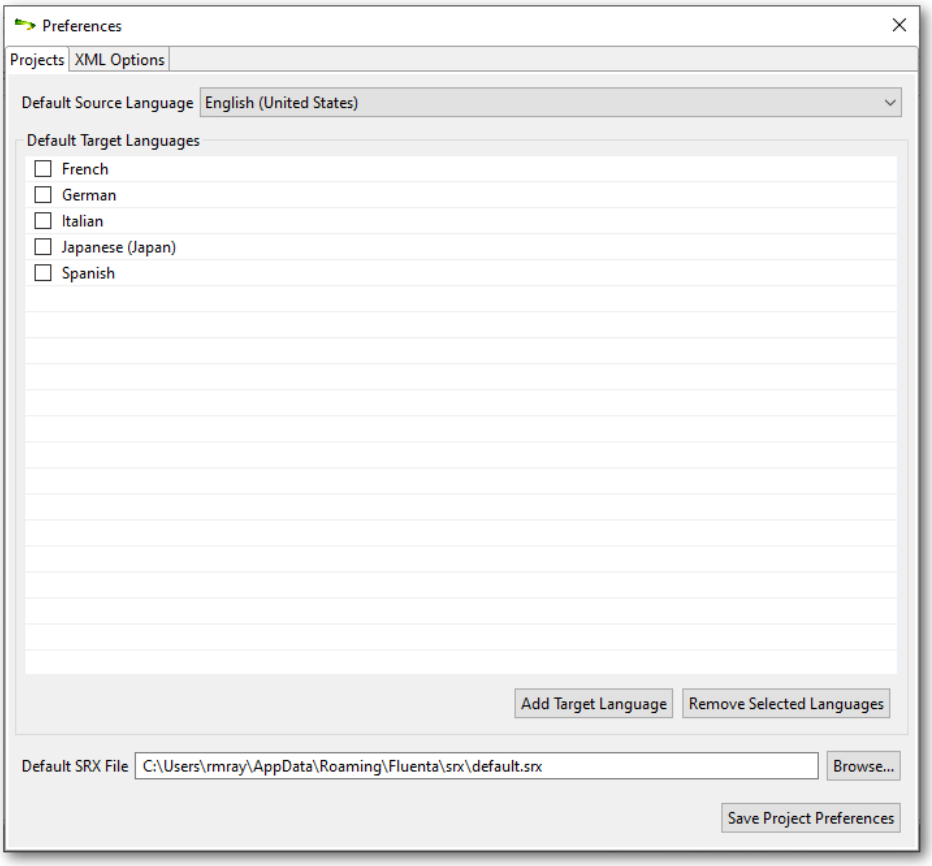
Advanced Configuration

Fluenta default settings can be changed in the **Preferences** dialog.

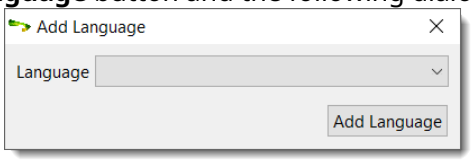
- To open the **Preferences** dialog on Windows or Linux, select **Preferences** in **Settings** menu.
- To open the **Preferences** dialog on Mac OS X, select **Preferences** in Apple menu.

Project Options

The following picture shows the **Projects** tab of the **Preferences** dialog.



- To add new target languages to the default set:
  1. click the **Add Target Language** button and the following dialog will appear:



  2. Select a language from the **Language** drop-down list.
  3. Click the **Add Language** button.

- To remove unnecessary languages from the default set:
  1. Select the check boxes next to each language you want to remove.
  2. Click the **Remove Selected Languages** button.

- To change the default SRX (Segmentation Rules eXchange) file that Fluenta uses for segmenting XLIFF files:
  1. Type the name of the SRX file in the **Default SRX File** text box or use the **Browse...** button to select an SRX file from the file system.

Click the **Save Project Preferences** button after making any change to the default settings.

XML Options

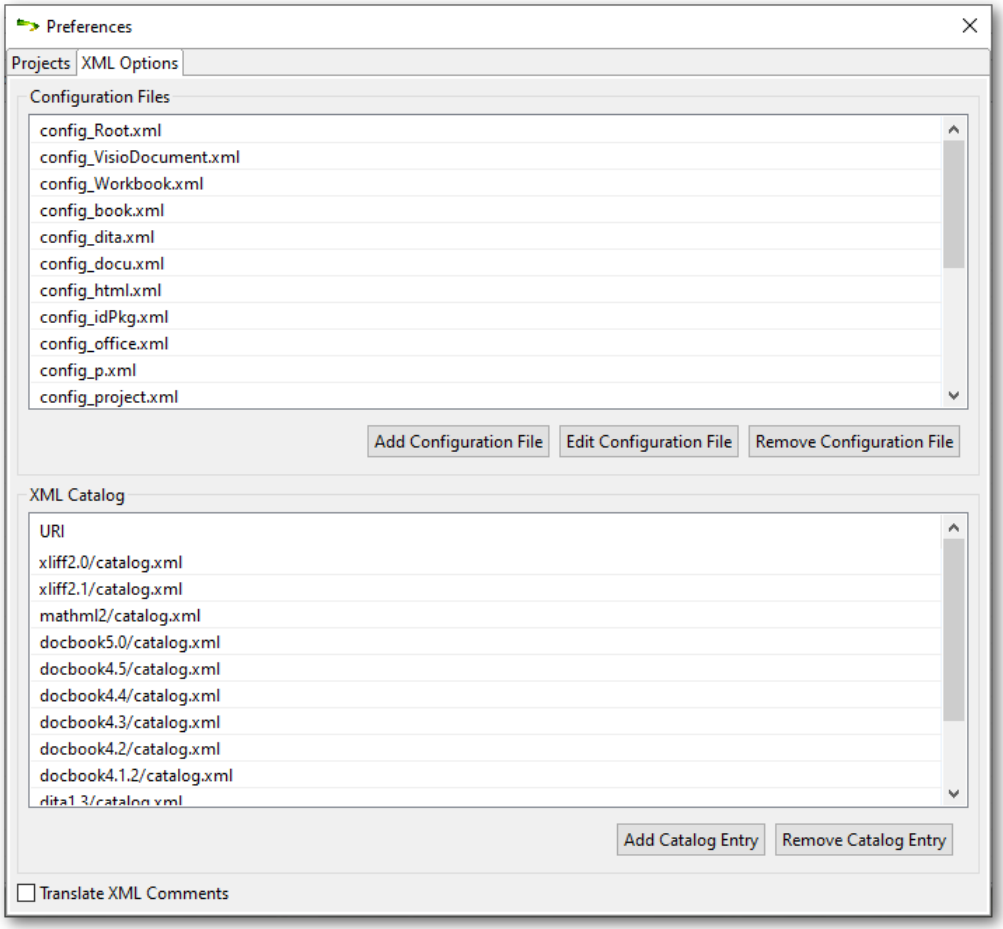
Fluenta needs to know two things for working with XML files:

- How to locate the grammar rules and entities declared in an XML file, if any.
- What elements and attributes contain translatable text.

XML catalogs that follow the specification published at http://www.oasis-open.org/committees/entity/spec-2001-08-06.html by OASIS are used to resolve the location of XML DTDs and Schemas.

Special XML files are used to configure the elements and attributes that contain translatable text. These files are used by the internal XML filter to extract text for processing. The configuration files are created and maintained using the application's graphical user interface.

The following picture shows the **XML Options** tab of the **Preferences** dialog:



XML Catalog

The application includes a default XML catalog with DTDs and XML Schemas for the most relevant formats and supported document types. Additional DTDs and XML Schemas can be added by the user as needed.

Select the **XML Catalog** tab in the **Preferences** dialog to add or remove entries from the catalog.

Add Catalog Entry

Click the **Add Catalog Entry** button and a file selection dialog will appear. Locate the catalog in the file system and select it.

Remove Catalog Entry

Select the catalog entry to remove in the catalogs table. Click the **Remove Catalog Entry** button and the application's catalog will be updated to reflect the change.

Configuration Files

Fluenta includes a configuration file for the 600+ elements defined defined in Appendix B.6 of DITA 1.3 All-Inclusive Edition.

A configuration file for Scalable Vector Graphics (SVG) is also shipped in fluenta installers.

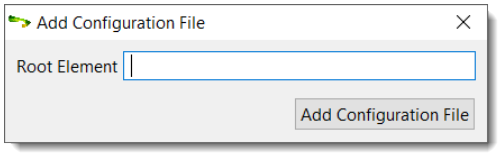
Additional configurations can be added by the user as required.

Add Configuration File

Procedure

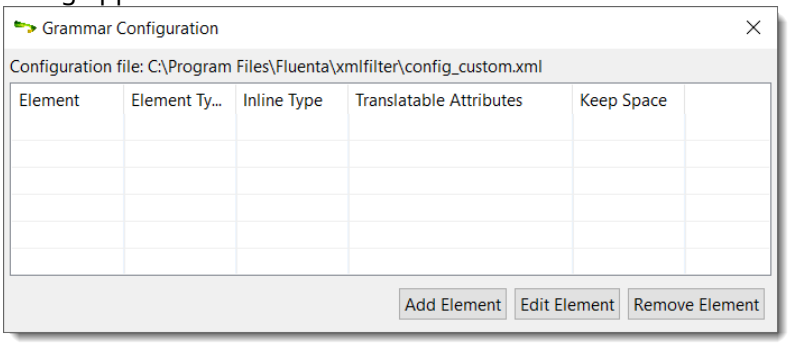
1. In **XML Options** tab of the **Preferences** dialog, click the **Add Configuration File** button.

   The following dialog appears:



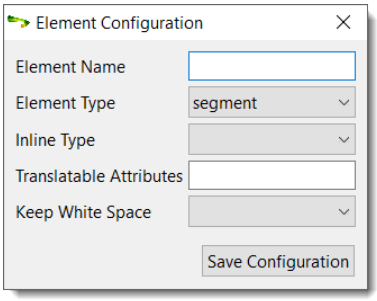
2. Type the name of the root element of your XML files in the **Root Elementt** text box. The name of the root element is used to name the configuration file.
3. Click the **Add Configuration File** button.

   The following dialog appears:



4. Click the **Add Element** button to add the configuration of an element.

   The following dialog appears:



5. Type the name of the element being added in the **Element Name** text box.
6. Select the type of element in the **Element Type** drop-down list. Available types are:
   - **segment**: the selected element starts a new section of translatable text.
   - **inline**: the selected element represents a change in formatting options and does not start a new section of translatable text.
   - **ignore**: the selected element and its children should be ignored.
7. If the element type is "inline", select the kind of formatting represented by the element in the **Inline Type** drop-down list.
8. If the element has translatable attributes, enter their names separated by a ";" in the **Translatable Attributes** text box.
9. If white space needs to be preserved when extracting text, select "Yes" in the **Keep White Space** drop-down list.
10. Click the **Save Configuration** button.
11. Repeat the previous steps until all required elements have been configured.

Results

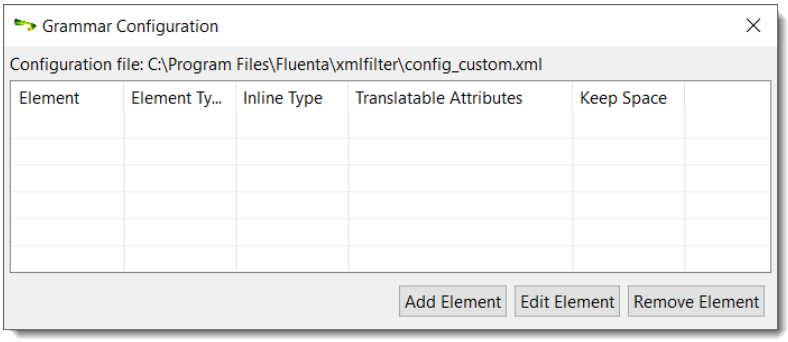
A new configuration file for the XML filter is created.

Edit Configuration File

Procedure

1. In **XML Options** tab of the **Preferences** dialog, select the configuration file to edit.
2. Click the **Edit Configuration File** button.

   The following dialog appears:



3. Use the buttons in the **Grammar Configuration** dialog to update the configuration file.
   - Use the **Add Element** button to add a new element to the configuration file.
   - Use the **Edit Element** button to modify the properties of an existing element.
   - Use the **Remove Element** button to delete an element from the configuration file.
4. Repeat the previous step until all elements are properly configured.

Remove Configuration File

Procedure

1. In **XML Options** tab of the **Preferences** dialog, select the configuration file to remove.
2. Click the **Remove Configuration File** button.

Results

Selected configuration file is removed and the list of configuration files is updated to reflect the change.

Subscriptions

Fluenta is available in two modes:

- Personal Use of Source Code
- Yearly Subscriptions

Personal Use of Source Code

Source code of Fluenta is free for personal use. Anyone can download the source code from Github.com, compile, modify and use it at no cost in compliance with the accompanying license terms.

Subscriptions

The version of Fluenta included in the official installers from Maxprograms Download Page can be used at no cost for 30 days requesting a free Evaluation Key.

Subscription Keys are available at Maxprograms Online Store. Subscription Keys cannot be shared or transferred to different machines.

Subscription version includes unlimited direct email support at tech@maxprograms.com

Differences Summary

	Source Code	Subscription Based
Ready To Use Installers	No	Yes
Notarized macOS launcher	No	Yes
Signed launcher and installer for Windows	No	Yes
Restricted Features	None	None
Technical Support	• Peer support at Groups.io	• Direct email at tech@maxprograms.com • Peer support at Groups.io

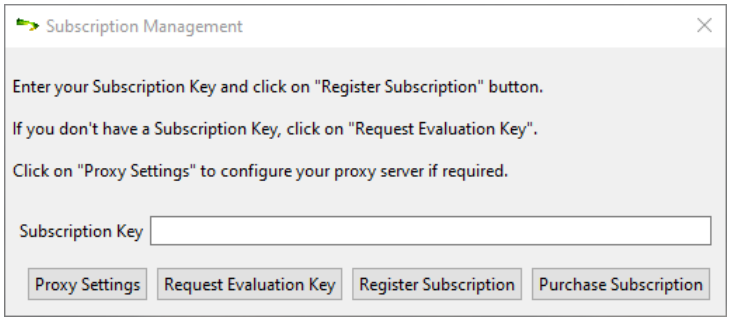
Request an Evaluation Subscription

About this task

You can evaluate the program for free during 30 days before purchasing a Subscription Key. All features are enabled during the evaluation period.

Procedure

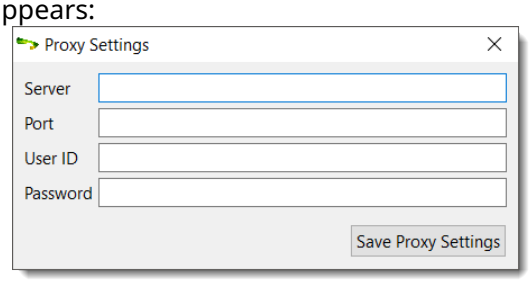
1. When you start the program for the first time, the following dialog appears:



2. If your computer uses a proxy server to connect to the Internet, follow these steps to configure the proxy server settings:

   a. Click the **Proxy Settings** button.

      The following dialog appears:



   b. Type the proxy server name or IP in the **Server** text box.

   c. Type the proxy port number in the **Port** text box.

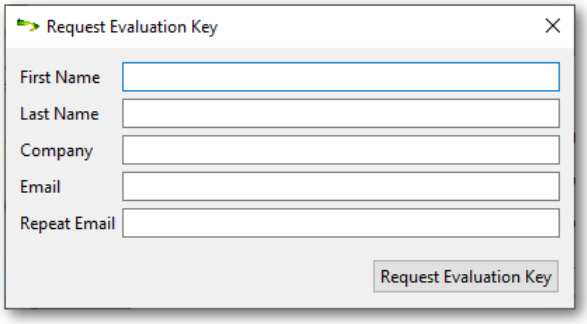
   d. If your proxy server requires authentication, type the proxy user name in the **User ID** text box and the corresponding password in the **Password** text box.

   e. Click the **Save Proxy Settings** button.

      Selected proxy settings are saved and the dialog closes.

3. Click the **Request Evaluation Key** button.

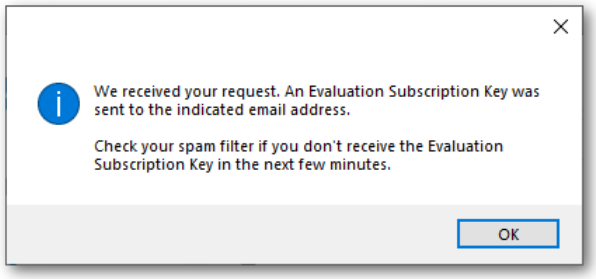
   The following dialog appears:



4. Type your first name in the **First Name** text box.
5. Type your last name in the **Last Name** text box.
6. Enter your company name in the **Company** text box. *This step is optional.*
7. Type your email address in the **Email** text box.

8. Enter again your email addres in the **Repeat Email** text box.
9. Click the **Request Evaluation Key** button.

   Your evaluation subscription request is sent to the Registration Server. An email with an evaluation subscription key will be immediately sent to the indicated email address.



10. Check your email and note the new evaluation subscription key. Check your spam filter if you don't receive an email with the evaluation subscription key within a few minutes.
11. Enter the evaluation subscription in the **Subscription Key** text box of the **Subscription Management** dialog.
12. Click the **Register Subscription** button.

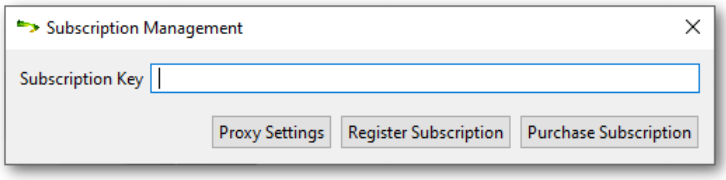
Results

Your computer is enabled to work with the application for 30 days.

Register a Subscription Key

Procedure

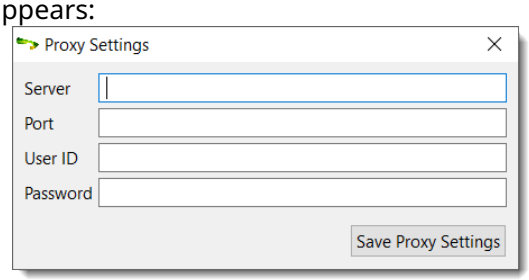
1. When you start the program after the trial period has ended, the following dialog appears:



2. Type your subscription code in the **Subscription Key** text box.
3. If your computer uses a proxy server to connect to the Internet, follow these steps to configure the proxy server settings:

   a. Click the **Proxy Settings** button.

      The following dialog appears:



b. Type the proxy server name or IP in the **Server** text box.

c. Type the proxy port number in the **Port** text box.

d. If your proxy server requires authentication, type the proxy user name in the **User ID** text box and the corresponding password in the **Password** text box.

e. Click the **Save Proxy Settings** button.

   Selected proxy settings are saved and the dialog closes.

4. Click the **Register Subscription** button.

Results

Your subscription key code is sent to the Registration Server and your computer is enabled to work with the registered application.

Command Line Interface

Fluenta can be integrated into publishing and editing workflows using its Command Line Interface (CLI).

The methods exposed by Fluenta's CLI are:

- Create a project;
- Remove project;
- Retrieve the list of projects;
- Generate XLIFF files and optional word counts;
- Import an XLIFF file;
- Create a memory;
- Retrieve the list of memories;
- Import a TMX file into a memory;
- Export a memory as TMX file;
- Register a Subscription Key.

On Mac OS X and Linux use **`fluenta_cli.sh`** for executing Fluenta without the graphical user interface.

- On Mac OS X the default location for the script is `/Applications/Fluenta/fluenta_cli.sh`
- On Linux the default location for the shell script is `/opt/Maxprograms/Fluenta/fluenta_cli.sh`

On Windows use **`fluenta_cli.bat`** (available in the default installation directory) to process command line parameters.

Create Project

Use the following parameters to create a project from command line:

Parameter	Value	Description
-add	data.json	A JSON file containing the data required for creating the project.

Example:

```
fluenta_cli.bat -add addProject.json
```

Where the content of **`addProject.json`** should be like:

```
{ id:12345678,
  title:"JSON test",
  description:"project created using JSON and CLI",
  map:"D:\\sample content\\en\\User_Guide-use-only.ditamap",
  srcLang:"en-US",
```

```
  tgtLang:["es","fr"]
}
```

JSON Value Pairs

Member	Data Type	Description
id	Number (long integer)	Project identifier
title	String	Short text description for the project. This is the description displayed in Fluenta's GUI.
description	String	A longer text description for the project.
map	String	Path to the project DITA map.
srcLang	String	Language code for the original DITA content.
tgtLang	String Array	An array containing the target languages for the project.
memories	Number Array	[Optional] An array containing the ids of existing memories to be associated with the project.

Remove Project

Use the following parameters to remove a project from command line:

Parameter	Value	Description
-del	projectId	Project identifier

Example:

```
fluenta_cli.bat -del 12345678
```

Retrieve Project List

Use the following parameters to retrieve the list of projects from command line:

Parameter	Value	Description
-getProjects	-	-

Example:

```
./fluenta_cli.sh -getProjects
```

Output example:

```
{"projects": [
   {
      "owner": "admin",
      "tgtLang": [
         "es",
         "fr"
      ],
      "lastUpdate": "2015-09-06 16:09",
      "memories": [12345678],
      "description": "project created using JSON and CLI",
      "id": 12345678,
      "title": "JSON test",
      "creationDate": "2015-09-05 08:29",
      "targetStatus": {
         "fr": "In Progress",
         "es": "Completed"
      },
      "map": "/Users/admin/sample content/en/User_Guide-use-only.ditamap",
      "srcLang": "en-US",
      "status": "In Progress"
   },
   {
      "owner": "admin",
      "tgtLang": ["fr"],
      "lastUpdate": "2015-07-29 18:50",
      "memories": [1438205821009],
      "description": "Fluenta Demo",
      "id": 1438205821009,
      "title": "Thunderbird",
      "creationDate": "2015-07-29 18:37",
      "targetStatus": {"fr": "In Progress"},
      "map": "/Users/admin/sample content/en/User_Guide.ditamap",
      "srcLang": "en-US",
      "status": "In Progress"
   }
]}
```

Generate XLIFF Files

Use the following parameters to generate XLIFF files for a project from command line:

Parameter	Value	Description
-generateXLIFF	data.json	A JSON file containing the data required for generating XLIFF files.
-verbose		[Optional] Selects whether progress information is logged or not in stdout.

Example:

```
fluenta_cli.bat -generateXLIFF genXLIFF.json -verbose
```

Where the content of **`genXLIFF.json`** should be like:

```
{ id:12345678,
  xliffFolder: "C:\\sample data\\XLIFF",
  ditaval: "C:\\sample data\\filter.ditaval",
  useICE: true,
  useTM: true,
  generateCount: false,
  useXLIFF20: false,
  tgtLang: ["es", "fr"]
}
```

JSON Value Pairs

Member	Data Type	Description
id	Number (long integer)	Project identifier
xliffFolder	String	Path to the folder wher XLIFF files and optional word counts should be stored.
ditaval	String	Path to a .ditaval file for filtering content to be extracted (this parameter is optional).
useICE	Boolean	Selects whetеher existing ICE matches should be reused.
useTM	Boolean	Selects whether translation memories should be used.
generateCount	Boolean	Selects whether word counts should be generated.
useXLIFF20	Boolean	Selects whether XLIFF 2.0 will be generated instead of XLIFF 1.2 (default).
tgtLang	String Array	An array containing the target languages for the XLIFF files.

Import XLIFF File

Use the following parameters to import a translated XLIFF file into a project from command line:

Parameter	Value	Description
-importXLIFF	data.json	A JSON file containing the data required for importing the XLIFF file.
-verbose		[Optional] Selects whether progress information is logged or not in stdout.

Example:

```
fluenta_cli.bat -importXLIFF impXLIFF.json -verbose
```

Where the content of **`impXLIFF.json`** should be like:

```
{ id:12345678,
  xliffFile: "C:\\sample data\\XLIFF\\spanish.xlf",
  outputFolder: "C:\\sample data\es\\",
  updateTM: true,
  acceptUnapproved: true,
  ignoreTagErrors: false,
  cleanAttributes: true
}
```

JSON Value Pairs

Member	Data Type	Description
id	Number (long integer)	Project identifier
xliffFile	String	Path to the XLIFF file to be imported.
outputFolder	Boolean	Path to the folder where the translated DITA files should be stored.
updateTM	Boolean	Selects whether the memory associated with the project should be updated with the translations in the XLIFF file.
acceptUnapproved	Boolean	Selects whether segments that are translated but not approved should be treated as approved.
ignoreTagErrors	Boolean	Selects whether inline tag errors should be ignored. If set to `false` and errors are found, a detailed report in HTML format is generated and stored in the folder that contains the XLIFF file.

Member	Data Type	Description
cleanAttributes	Boolean	Selects whether the default values for attributes `@class`, `@xmlns:ditaarch`, `@ditaarch:DITAArchVersion` and `@domains` that are automatically added by XML parsers should be removed in translated DITA files.

Create Memory

Use the following parameters to create a memory from command line:

Parameter	Value	Description
-addMem	data.json	A JSON file containing the data required for creating the memory.

Example:

```
fluenta_cli.bat -addMem addMemory.json
```

Where the content of **`addMemory.json`** should be like:

```
{ id:12345678,
  title:"JSON test",
  description:"memory created using JSON and CLI",
  srcLang:"en-US",
  tgtLang:["es","fr"]
}
```

JSON Value Pairs

Member	Data Type	Description
id	Number (long integer)	Memory identifier
title	String	Short text description for the memory. This is the description displayed in Fluenta's GUI.
description	String	A longer text description for the memory.
srcLang	String	Source language code.
tgtLang	String Array	An array containing the target languages for the memory.

Retrieve Memory List

Use the following parameters to retrieve the list of memories from command line:

Parameter	Value	Description
-getMemories	-	-

Example:

```
./fluenta_cli.sh -getMemories
```

Output example:

```
{"memories": [
   {
      "owner": "admin",
      "tgtLang": [
         "es",
         "fr"
      ],
      "lastUpdate": "",
      "name": "JSON test",
      "description": "project created using JSON and CLI",
      "id": 12345678,
      "creationDate": "2015-09-05 08:29",
      "srcLang": "en-US"
   },
   {
      "owner": "admin",
      "tgtLang": [],
      "lastUpdate": "2015-07-29 18:44",
      "name": "Thunderbird",
      "description": "Fluenta Demo",
      "id": 1438205821009,
      "creationDate": "2015-07-29 18:37",
      "srcLang": "en-US"
   }
]}
```

Import TMX File

Use the following parameters to import a TMX file into a memory from command line:

Parameter	Value	Description
-importTmx	memId	The id of the memory.
-tmx	tmxFile	Path to the TMX file to be imported

Parameter	Value	Description
-verbose		[Optional] Selects whether progress information is logged or not in stdout.

Example:

```
fluenta_cli.bat -importTmx 12345678 -tmx "c:\sample data\updated.tmx" -verbose
```

Export TMX File

Use the following parameters to export a memory as a TMX file from command line:

Parameter	Value	Description
-exportTmx	memId	The id of the memory to be exported.
-tmx	tmxFile	Path to the TMX file to be generated.

Example:

```
fluenta_cli.bat -exportTmx 12345678 -tmx "c:\sample data\exported.tmx"
```

Subscription Management

Use the following parameters to register a Subscription Key from command line:

Parameter	Value	Description
-reg	key	The Subscription Key to be registered.

Example:

```
fluenta_cli.bat -reg HAL9000
```

Note

Registering a Subscription Key requires an Internet connection.

Java API

Fluenta can be integrated as a Java library in DITA related tools like Content Management Systems (CMS) or custom publishing engines.

All jar files included in `/lib` folder of Fluenta must be included in the CLASSPATH variable of the enclosing Java application.

The class `com.maxprograms.fluenta.API` exposes the following static methods:

- `public static void addProject(long id, String title, String description, String map, String srcLang, String[] tgtLang, long[] memIds)`
- `public static void removeProject(long id)`
- `public static String getProjects()`
- `public static void generateXLIFF(long id, String xliffFolder, String[] tgtLang, boolean useICE, boolean useTM, boolean generateCount, boolean verbose, String ditaval, boolean useXliff20) throws IOException`
- `public static void importXLIFF(long id, String xliffFile, String outputFolder, boolean updateTM, boolean acceptUnapproved, boolean ignoreTagErrors, boolean cleanAttributes, boolean verbose) throws IOException`
- `public static void addMemory(long id, String title, String description, String srcLang, String[] tgtLang)`
- `public static String getMemories()`
- `public static void importMemory(long id, String tmxFile, boolean verbose)`
- `public static void exportMemory(long id, String tmxFile) throws Exception`
- `public static String registerLicense(String licenseKey)`
- `public static String disableLicense(String licenseKey)`

Create Project

Use the static `addProject()` method exposed by class `com.maxprograms.fluenta.API` to create a project.

Method

```
public static void addProject(long id, String title, String description, String map, String srcLang, String[] tgtLang, long[] memIds) throws IOException
```

Parameters

Type	Name	Description
long	id	Project identifier
String	title	Short text description for the project. This is the description displayed in Fluenta's GUI.

Type	Name	Description
String	description	A longer text description for the project.
String	map	Path to the project DITA map.
String	srcLang	Language code for the original DITA content.
String[]	tgtLang	An array containing the target languages for the project.
long[]	memIds	An array containing the ids of existing memories to be associated with the project.

Returns

Nothing.

A project and an associated memory with same id and description are created on success.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...

long id = 123456;
String title = "test project";
String description = "sample project created using Java API";
String map = "/home/data/mainMap.ditamap";
String srcLang = "en-US";
String[] tgtLang = {"fr", "de", "ja"};
long[] memIds = {2345, 8547, 7852};

try{
   API.addProject( id, title, description, map, srcLang, tgtLang, memIds);
} catch (IOException ioe) {
   ioe.printStackTrace();
}
```

Remove Project

Use the static `removeProject()` method exposed by class `com.maxprograms.fluenta.API` to create a project.

Method

```
public static void removeProject(long id) throws IOException
```

Parameters

Type	Name	Description
long	id	Project identifier

Returns

Nothing.

A project and an associated memory with same id and description are created on success.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...

long id = 123456;

try{
  API.removeProject( id );
} catch (IOException ioe) {
  ioe.printStackTrace();
}
```

Retrieve Project List

Use the static `getProjects()` method exposed by class `com.maxprograms.fluenta.API` to retrieve the list of projects in JSON format.

Method

```
public static String getProjects() throws IOException
```

Parameters

None.

Returns

A `String` containing a JSON object.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...
try {
  String result = API.getProjects();
  System.out.println(result);
} catch (IOException ioe) {
   ioe.printStackTrace();
}
```

Output example:

```
{"projects": [
   {
      "owner": "admin",
      "tgtLang": [
         "es",
         "fr"
      ],
      "lastUpdate": "2015-09-06 16:09",
      "memories": [12345678],
      "description": "project created using JSON and CLI",
      "id": 12345678,
      "title": "JSON test",
      "creationDate": "2015-09-05 08:29",
      "targetStatus": {
         "fr": "In Progress",
         "es": "Completed"
      },
      "map": "/Users/admin/sample content/en/User_Guide-use-only.ditamap",
      "srcLang": "en-US",
      "status": "In Progress"
   },
   {
      "owner": "admin",
      "tgtLang": ["fr"],
      "lastUpdate": "2015-07-29 18:50",
      "memories": [1438205821009],
      "description": "Fluenta Demo",
      "id": 1438205821009,
      "title": "Thunderbird",
      "creationDate": "2015-07-29 18:37",
      "targetStatus": {"fr": "In Progress"},
      "map": "/Users/admin/sample content/en/User_Guide.ditamap",
      "srcLang": "en-US",
      "status": "In Progress"
   }
```

```
]}
```

Generate XLIFF Files

Use the static `generateXLIFF()` method exposed by class `com.maxprograms.fluenta.API` to generate XLIFF files and optional word counts for a project.

Method

```
public static void generateXLIFF(long id, String xliffFolder, String[] tgtLang, boolean useICE, boolean useTM, boolean generateCount, boolean verbose, String ditaval, boolean useXliff20) throws IOException
```

Parameters

Type	Name	Description
long	id	Project identifier
String	xliffFolder	Path to the folder wher XLIFF files and optional word counts should be stored.
String[]	tgtLang	An array containing the target languages for the XLIFF files.
boolean	useICE	Selects wheteher existing ICE matches should be reused.
boolean	useTM	Selects whether translation memories should be used.
boolean	generateCount	Selects whether word counts should be generated.
boolean	verbose	Selects whether progress information is logged or not in stdout.
String	ditaval	Path to a .ditaval containing filtering content to be extracted. This paramater may be null or empty if conditional processing is not required.
boolean	useXliff20	Selects whether XLIFF 2.0 will be generated instead of XLIFF 1.2 (default).

Returns

Nothing.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...

long id = 123456;
String xliffFolder = "/home/data/XLIFF";
String ditaval = "/home/data/filter.ditaval"; // can be null if not needed
String[] tgtLang = {"fr", "de", "ja"};
long[] memIds = {2345, 8547, 7852};
boolean useICE = true;
boolean useTM = false;
boolean generateCount = true;
boolean verbose = true;
boolean useXliff20 = true;
try{
  API.generateXLIFF(id, xliffFolder, tgtLang,  useICE, useTM, generateCount,
     verbose, ditaval, useXliff20);
} catch (IOException ioe) {
  ioe.printStackTrace();
}
```

Import XLIFF File

Use the static `importXLIFF()` method exposed by class `com.maxprograms.fluenta.API` to import a translated XLIFF file into an existing project.

Method

`public static void importXLIFF(long id, String xliffFile, String outputFolder, boolean updateTM, boolean acceptUnapproved, boolean ignoreTagErrors, boolean cleanAttributes, boolean verbose) throws IOException`

Parameters

Type	Name	Description
long	id	Project identifier
String	xliffFile	Path to the XLIFF file to be imported.
String	outputFolder	Path to the folder where the translated DITA files should be stored.
boolean	updateTM	Selects wheteher the memory associated with the project should be updated with the translations in the XLIFF file.

Type	Name	Description
boolean	acceptUnapproved	Selects whether segments that are translated but not approved should be treated as approved.
boolean	ignoreTagErrors	Selects whether inline tag errors should be ignored. If set to `false` and errors are found, a detailed report in HTML format is generated and stored in the folder that contains the XLIFF file.
boolean	cleanAttributes	Selects whether the default values for attributes `@class`, `@xmlns:ditaarch`, `@ditaarch:DITAArchVersion` and `@domains` that are automatically added by XML parsers should be removed in translated DITA files.
boolean	verbose	Selects whether progress information is logged or not in stdout.

Returns

Nothing.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...

long id = 123456;
String xliffFile = "/home/data/XLIFF/spanish.xlf";
String outputFolder = "/home/data/es/";
boolean updateTM = true;
boolean acceptUnapproved = true;
boolean ignoreTagErrors = false;
boolean cleanAttributes = true;
boolean verbose = true;
try{
  API.importXLIFF( id, xliffFile, outputFolder,  updateTM, acceptUnapproved,
                   ignoreTagErrors, cleanAttributes, verbose);
} catch (IOException ioe) {
  ioe.printStackTrace();
}
```

Create Memory

Use the static `addMemory()` method exposed by class `com.maxprograms.fluenta.API` to create a memory.

Method

`public static void addMemory(long id, String title, String description, String srcLang, String[] tgtLang) throws IOException`

Parameters

Type	Name	Description
long	id	Memory identifier
String	title	Short text description for the memory. This is the description displayed in Fluenta's GUI.
String	description	A longer text description for the memory.
String	srcLang	Source language code.
String[]	tgtLang	An array containing the target languages for the memory.

Returns

Nothing.

A memory is created on success.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...

long id = 123456;
String title = "test memory";
String description = "sample memory created using Java API";
String srcLang = "en-US";
String[] tgtLang = {"fr", "de", "ja"};

try{
   API.addMemory( id,  title, description, srcLang, tgtLang);
} catch (IOException ioe) {
   ioe.printStackTrace();
```

```
}
```

Retrieve Memory List

Use the static `getMemories()` method exposed by class `com.maxprograms.fluenta.API` to retrieve the list of memories in JSON format.

Method

```
public static String getMemories() throws IOException
```

Parameters

None.

Returns

A `String` containing a JSON object.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...
try {
  String result = API.getMemories();
  System.out.println(result);
} catch (IOException ioe) {
  ioe.printStackTrace();
}
```

Output example:

```
{"memories": [
   {
      "owner": "admin",
      "tgtLang": [
         "es",
         "fr"
      ],
      "lastUpdate": "",
      "name": "JSON test",
      "description": "project created using JSON and CLI",
      "id": 12345678,
      "creationDate": "2015-09-05 08:29",
      "srcLang": "en-US"
```

```
   },
   {
      "owner": "admin",
      "tgtLang": [],
      "lastUpdate": "2015-07-29 18:44",
      "name": "Thunderbird",
      "description": "Fluenta Demo",
      "id": 1438205821009,
      "creationDate": "2015-07-29 18:37",
      "srcLang": "en-US"
   }
]}
```

Import Memory

Use the static `importMemory()` method exposed by class `com.maxprograms.fluenta.API` to import a TMX file into an existing memory.

Method

`public static void importMemory(long id, String tmxFile, boolean verbose) throws IOException`

Parameters

Type	Name	Description
long	id	Memory identifier
String	tmxFile	Path to the TMX file to be imported.
boolean	verbose	Selects whether progress information is logged or not in stdout.

Returns

Nothing.

Throws

`java.io.IOException` is thrown on error.

Example

```
import java.io.IOException;
import com.maxprograms.fluenta.API;
...
long id = 123456;
String tmxFile = "/opt/data/updates.tmx";
bolean logging = false;

try{
```

```
   API.importMemory( id,  tmxFile, logging);
} catch (IOException e) {
   e.printStackTrace();
}
```

Export Memory

Use the static `exportMemory()` method exposed by class `com.maxprograms.fluenta.API` to export a memory as a TMX file.

Method

`public static void exportMemory(long id, String tmxFile) throws Exception`

Parameters

Type	Name	Description
long	id	The id of the memory to be exported.
String	tmxFile	Path to the TMX file to be generated.

Returns

Nothing.

Throws

`java.lang.Exception` is thrown on error.

Example

```
import com.maxprograms.fluenta.API;
...
long id = 123456;
String tmxFile = "/opt/data/exported.tmx";

try{
   API.exportMemory( id,  tmxFile);
} catch (Exception e) {
   e.printStackTrace();
}
```

Subscription Management

Register Subscription Key

Use the static `registerLicense()` method exposed by class `com.maxprograms.fluenta.API` to register a Subscription Key.

Method

```
public static String registerLicense(String licenseKey)
```

Parameters

Type	Name	Description
String	licenseKey	The Subscription Key to be registered.

Returns

On success: returns the String "OK".

On error: returns a String object containing an error message.

Throws

Nothing.

Example

```
import com.maxprograms.fluenta.API;
...
String key = "HAL9000";
String result = API.registerLicense(key);
if (!result.equals("OK") {
   System.err.println("Error registering subscription: " + result);
   return;
}
```

Note

Registering a Subscription Key requires an Internet connection.

Glossary

JSON

JavaScript Object Notation (JSON) is an open standard for the serialization of structured data in text format . It is derived from the object literals of JavaScript, as defined in the ECMAScript Programming Language Standard, Third Edition. JSON specification is available at https://tools.ietf.org/html/rfc7159.

Language Service Provider (LSP)

A company or individual specialized in providing translation and localization services.

OASIS

OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society.

SRX

Segmentation Rules eXchange (SRX) is an XML-based open standard, originally published by LISA (Localization Industry Standards Association), for describing how translation and other language-processing tools segment text for processing.

TMX

Translation Memory eXchange (TMX) is an open standard originally published by LISA (Localization Industry Standards Association). The purpose of TMX is to allow easier exchange of translation memory data between tools and/or translation vendors with little or no loss of critical data during the process.

Translation Memory

Translation Memory (TM) is a language technology that enables the translation of segments (paragraphs, sentences or phrases) of documents by searching for similar segments in a database and suggesting matches that are found in the databases as possible translations.

XLIFF

XLIFF (XML Localization Interchange File Format) is an open standard developed by OASIS (Organization for the Advancement of Structured Information Standards). The purpose of this vocabulary is to store localizable data and carry it from one step of the localization process to the other, while allowing interoperability between tools.