

_Complexer task import example

- Create and upload task (POST example with fileupload) but do not start import directly
 - Get a list of language resources applicable to the uploaded task
 - Associate found language resources to the task
 - Schedule and configure analysis and pre-translation
 - Change other task configuration (TO BE DOCUMENTED)
- Final start of the import
- Task import callback



Please see Basic Usage Example before! Authentication and task listing is for example shown there.

Create and upload task (POST example with fileupload) but do not start import directly

Due historical reasons coming from the used frontend framework, POST requests containing file uploads are different as the other POST requests without files.

[More information about the task API.](#)



A POST with fileupload contains the entities attributes not as JSON, but as plain form fields. This is a known caveat.

Please change also the XLF file path to the file which should be uploaded.

Curl Example

```
curl 'https://www.translate5.net/editor/task' -X POST \
-H 'Translate5AuthToken: YOUR_APP_TOKEN' \
-F "format=jsontext" \
-F "taskName=This is a import test" \
-F "sourceLang=de" \
-F "targetLang=it" \
-F "edit100PercentMatch=1" \
-F "lockLocked=1" \
-F "autoStartImport=0" \
-F "importUpload=@/PATH/TO/YOUR/FILE.xlf"
```

On success this results in a JSON containing the importing task.

The autoStartImport=0 is the most important difference to a normal import here, it prevents the direct import processing of the upload task and enables the following steps to configure for example the match analysis.

Get a list of language resources applicable to the uploaded task

As next step you should get the language resources applicable to the uploaded task. In the following request URL replace %7B_THE_TASK_GUID_%7D with the taskGuid generated by the above request. Where %7B and %7D are the encoded curly braces of the taskGuid and should remain in the URL.

The following request provides a list of language resources which can be used for the newly created task.

```
curl 'http://translate5.localdev/editor/languageresourcetaskassoc?_dc=1621243689215&filter=%5B%7B%22property%22%3A%22taskGuid%22%2C%22operator%22%3A%22eq%22%2C%22value%22%3A%22%7B_THE_TASK_GUID_%7D%22%7D%5D' \
-H 'Connection: keep-alive' \
-H 'Accept: application/json' \
-H 'Translate5AuthToken: YOUR_APP_TOKEN'
```

Associate found language resources to the task

To associate a language resource create a new languageresourcetaskassoc entry by posting the following JSON to the according URL as exemplified below:

JSON data to be POSTed

```
{
  "languageResourceId":"2614",    // the numeric ID of the language resource provided by the above GET
  "taskGuid":"{928ac493-618e-4475-9436-521c2e3432e7}", //the taskGuid of the created task
  "segmentsUpdateable":false      // true only usable for TMs which then are getting updated on editing the
  segments
}
```

as CURL example

```
curl 'http://translate5.localdev/editor/languageResourceTaskAssoc' \
-X POST \
-H 'Connection: keep-alive' \
-H 'Accept: application/json' \
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8' \
-H 'Translate5AuthToken: YOUR_APP_TOKEN'
--data-raw 'data=%7B%22languageResourceId%22%3A%222614%22%2C%22taskGuid%22%3A%22%7B928ac493-618e-4475-9436-521c2e3432e7%7D%22%2C%22segmentsUpdateable%22%3Afalse%7D'
```

Schedule and configure analysis and pre-translation

```
curl 'http://translate5.localdev/editor/task/9015/pretranslation/operation' \
-X 'PUT' \
-H 'Connection: keep-alive' \
-H 'Accept: application/json' \
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8' \
-H 'Translate5AuthToken: YOUR_APP_TOKEN'
--data-raw 'internalFuzzy=0&pretranslateMatchrate=100&pretranslateTmAndTerm=1&pretranslateMt=1&termtaggerSegment=0&isTaskImport=1&batchQuery=0'
```

The used parameters control the analysis and pre-translation behaviour as documented in [Plug-In Matchanalysis#injectedtaskoperationtostarttheanalysisandpre-translation](#).

Change other task configuration (TO BE DOCUMENTED)

The following call provides a list of configuration parameters applicable for the newly created task.

```
curl 'http://translate5.localdev/editor/config?_dc=1621243733095&taskGuid=%7B928ac493-618e-4475-9436-521c2e3432e7%7D' \
-H 'Connection: keep-alive' \
-H 'Accept: application/json' \
-H 'Translate5AuthToken: YOUR_APP_TOKEN'
```

Specific configurations for a specific task import can be set like this:

```
curl 'http://translate5.localdev/editor/config/runtimeOptions.import.csv.fields.source' \
-X 'PUT' \
-H 'Connection: keep-alive' \
-H 'Accept: application/json' \
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8' \
-H 'Translate5AuthToken: YOUR_APP_TOKEN'
--data-raw 'taskGuid=%7B32309a4f-6a22-488a-ac50-b8b520896bdb%7D&data=%7B%22value%22%3A%22source123%22%2C%22taskGuid%22%3A%22%7B32309a4f-6a22-488a-ac50-b8b520896bdb%7D%22%2C%22name%22%3A%22runtimeOptions.import.csv.fields.source%22%7D'
```

Where "runtimeOptions.import.csv.fields.source" in the URL has to be replaced with the config name to be set, and the values and taskGuid in the raw data has to be adjusted too. Raw data JSON:

```
{
  "value":"source123",    // the new value to be used for the configuration
  "taskGuid":"{32309a4f-6a22-488a-ac50-b8b520896bdb}", // the task guid of the task for which the config
  // should be set
  "name":"runtimeOptions.import.csv.fields.source"      // the name of the config again
}
```

Final start of the import

Finally the most important step is a GET call to the tasks import URL to start the previously configured import, where NUMERIC_TASK_ID has to be replaced with the numeric task ID.

```
curl 'http://translate5.localdev/editor/task/NUMERIC_TASK_ID/import' \
-H 'Connection: keep-alive' \
-H 'Accept: application/json' \
-H 'Translate5AuthToken: YOUR_APP_TOKEN'
```

Task import callback

Since a import is running in the background, the so created task will be returned with a state = import. For checking if the import is done, a callback should be configured: [task import push callback](#)