translate5 TM service - REST API

In this document the translate5 TM service REST interface is described.

The translate5 TM service is build by using the OpenTM2 Translation Memory Engine.

It provides the following functionality:

- import new openTM2-TMs
- delete openTM2-TMs
- create new empty openTM2-TM
- import TMX
- open TM and close TM: not possible see extra section in this document.
- query TM for Matches: one query per TM, not quering multiple TMs at once.
- query TM for concordance search
- save new entry to TM

This can be achieved by the following specification of a RESTful HTTP Serive, the specification is given in the following form:

1. URL of the HTTP Resource, where servername and an optional path prefix is configurable.
2. HTTP Method with affected functionality
3. Brief Description
4. Sent and returned Body.

Request Data Format:

The transferred data in the requests is JSON, and is directly done in the request body.

URL Format:

In this document the OpenTM2 is always assumed under http://opentm2/.

To rely on full networking features (proxying etc.) the URL is configurable in Translate5, so that the OpenTM2 instance can also reside under: http://xyz/foo/bar/.

Errors

For each request the possible errors are listed below for each resource. In case of an error the body should contain at least the following JSON, if it is senseful the attributes of the original representation can be added.

{

errors: [{errorMsg: 'Given tmxData is no TMX.'}]

}

http://opentm2/translationmemory/

POST - creating a new or importing an existing filebased binary OpenTM2 TM

The Parameter „name“ contains the TM Name as a string. The string has a maxlength of 256 chars. It can contain any characters except the characters backslash (\), slash(/), colon (:), question mark (?), asterisk (*), vertical line (|), less than sign (<), and greater than sign (>).

Uploading a file is optional, omitting a file means creating a empty TM only.

If an empty TM is created, the POST request contains only the JSON structure with the TM Name.

If an existing binary OpenTM2 file should be additionally imported to the new TM, the POST must be encoded as multipart/form-data.

The JSON structure with the meta data will then be in the first chunk of the multiparted request, the chunk must be named “meta”.

The second chunk contains the plain binary file content and must be named “data”. This binary data contains the TM content

The resulting body contains the name of the TM, as given in the POST request.

LOGGING:

You can optionally set here or in other requests with JSON body logging level that could impact performance.

You can set the next levels:

- 0 - DEVELOP - could make code work really slow, should be used only when debugging some specific places in code, like binary search in files, etc.
- 1 - DEBUG - logging values of variables
- 2 - INFO - logging top-level functions entrances, return codes, etc.
- 3 - WARNING - logging if we reached some commented or hardcoded code. Usually commented code here is replaced with new code, and if not, it's marked as ERROR level
- 4 - ERROR - errors, why and where something fails during parsing, search, etc
- 5 - FATAL - you shouldn't reach this code, something is really wrongOther values would be ignored. The set level would stay the same till you change it in a new request or close the app. Logs suppose to be written into a file with date\time name under ~/.OtmMemoryService/Logs and errors/fatal are supposed to be duplicated in another log file with *FATAL* suffices

To OpenTM2 – without data / creating an empty TM:

```
{
sourceLang: “en”, // the source language is required for a new TM
name: „TM Name“,
[loggingThreshold:"2"]
}
```

Raw POST to OpenTM2 – with provided import file:

```
POST http://opentm2/translationmemory HTTP/1.1
Content-Type: multipart/form-data; boundary="autogenerated"

-- autogenerated
Content-Type: application/json; charset=utf-8
Content-Disposition: form-data; name=meta

{"name":"TM Name", sourceLang:"en"}

--autogenerated
Content-Type: image/jpeg
Content-Disposition: form-data; name=data; filename=Original Filename.jpg
...TM content ...
--autogenerated--
```

In both cases from OpenTM2 - HTTP 200 OK:

{

name: „TM Name“

}

Errors:

- 400 Bad Request – if parameters are missing or are not well formed.
- 409 Conflict – if a memory with the given name already exists.
- 500 Server Error – for other technical problems.

http://opentm2/translationmemory/[TM_Name]/import

POST import a TMX file into an existing OpenTM2 TM

To OpenTM2:

multipart/form-data like on POST above, expect that no separate JSON section is needed here.

Call answers directly after the upload is done, but before the import starts with HTTP 201 – this means: Import is created and will be started now.

From OpenTM2 - HTTP 201 OK:

{ // empty JSON object, since no data expected as result here!

}

Errors:

- 400 Bad Request – if parameters are missing or are not well formed.
- 404 Not Found – if the memory of the given name does not exist
- 500 Server Error – for other technical problems.

http://opentm2/translationmemory/[TM_Name]/status

GET status of a TM

To OpenTM2:

multipart/form-data like on POST above, expect that no separate JSON section is needed here.

From OpenTM2 - HTTP 200 OK:

{

'status':'import' //allowed status values: import, available, error

}

Errors:

- 400 Bad Request – if parameters are missing or are not well formed.
- 404 Not Found – if the memory of the given name does not exist
- 500 Server Error – for other technical problems.

http://opentm2/translationmemory/

GET – retrieving a list of available TM Files

To OpenTM2: -

From OpenTM2 - HTTP 200 OK:

[{

name: 'my nice TM'

}]

Errors:

- 500 Server Error – for other technical problems.

http://opentm2/translationmemory/[TM_Name]/

TM_Name is URL-encoded

GET – retrieving a single TM File

To OpenTM2: -

From OpenTM2 - HTTP 200 OK:

Same as POST from OpenTM2 result.

Errors:

- 404 Not Found – if TM file to given [TMID] in URL was not found
- 500 Server Error – for other technical problems.

DELETE – deletes an existing TM File

Adressed by the given URL, no body needed.

Errors:

- 404 Not Found – if TM file to given [TMID] in URL was not found
- 500 Server Error – for other technical problems.

PUT – updating an existing TM File in one request

Currently not needed, would be only to change the TM name

GET – list of all segments from TM

Currently not needed.

Opening and closing a TM

In first concept it was planned to implement routines to open and close a TM. While concepting we found some problemes with this approach:

- First one is the realization: opening and closing a TM by REST would mean to update the TM Resource and set a state to open or close. This is very awkward.
- Since in translate5 multiple tasks can be used to the same time, multiple tasks try to access one TM. Closing TMs is getting complicated to prevent race conditions in TM usage.
- Since OpenTM2 loads the whole TM in memory, OpenTM2 must control itself which TMs are loaded or not.

This leads to the following conclusion in implementation of opening and closing of TMs:

OpenTM2 has to automatically load the requested TMs if requested. Also OpenTM2 has to close the TMs after a TM was not used for some time. That means that OpenTM2 has to track the timestamps when a TM was last requested.

http://opentm2/translationmemory/[TM_Name]/openHandle

GET – Opens a memory for queries by OpenTM2

Note: This method is not required as memories are automatically opened when they are accessed for the first time.

http://opentm2/translationmemory/[TM_Name]/openHandle

DELETE – Closes a memory for queries by OpenTM2

Note: This method is not required as memories are automatically opened when they are accessed for the first time.

http://opentm2/translationmemory/[TM_Name]/entry/

POST – creates a new entry or updates target entry if match pair already exists

This method updates an existing proposal when a proposal with the same key information (source text, language, segment number, and document name) exists.

Parameters sourceLang and targetLang are containing the languages as RFC5646.

Parameters source and target are containing the entry contents to be stored. Format? plain string?

Attribute Parameters:

- documentName: contains the filename where the segment resides in Translate5.
- segmentNumber: evaluates to Translate5 segment mid.
- markupTable: OpenTM2 gets a new markup table named „translate5“, so this is the value which is delivered by Translate5.
- timestamp: this parameter is not set by translate5, but calculated automatically and delivered from OpenTM2 to translate5.
- author: contains the named user which provides the update / new entry
- In addition there are the following OpenTM2 Attributes currently not used by translate5:
    - context
    - additional info
    - type

To OpenTM2:

{

sourceLang: 'de',

targetLang: 'en',

source: „Das ist das Haus des Nikolaus“,

target: „This is the house of St. Nicholas“,

documentName: 'my file.sdlxliff',

segmentNumber: 123,

markupTable: 'translate5',

author: „Thomas Lauria“,

type: '',

timeStamp: '',

context: '',

addInfo: '',

[loggingThreshold:"2"]

}

The result from the server contains the same data as posted to the server. No additonal ID is added, since the entries are identified by the whole source string instead by an ID, only the timestamp is added.

From OpenTM2 – HTTP 200 OK:

{

sourceLang: 'de',

targetLang: 'en',

source: „Das ist das Haus des Nikolaus“,

target: „This is the house of St. Nicholas“,

documentName: 'my file.sdlxliff',

segmentNumber: 123,

markupTable: 'translate5',

timestamp: '2015-05-12 13:46:12',

author: „Thomas Lauria“

}

Errors:

- 404 Not Found – if TM file to given [TM_Name] in URL was not found
- 500 Server Error – for other technical problems.
- 400 Bad Request – if JSON parameters are missing or are not well formed.

http://opentm2/translationmemory/[TM_Name]/fuzzysearch/

POST– Serves a memory lookup based on the provided search criteria

To OpenTM2:

{

sourceLang: 'de',

targetLang: 'en-US',

source: „Das ist das Haus des Nikolaus“,

documentName: 'my file.sdlxliff', // can be empty

segmentNumber: 123, // can be empty

markupTable: 'translate5', // can be empty

context: „xyz“, // can be empty

[loggingThreshold:"2"]

}

From OpenTM2 HTTP 200 OK:

{

'NumOfFoundProposals': 2,

'results':

[{

source: „Das ist das Haus des Nikolaus“,

target: „This is the house of St. Nicholas“,

sourceLang: 'de', rfc5646

targetLang: 'en', rfc5646

matchRate: '100',

documentName: 'my file.sdlxliff',

DocumentShortName: 'shortnam.txt',

id: 'identifier',

type: 'Manual',

matchType: 'Exact',

segmentNumber: 123,

markupTable: 'XYZ',

timestamp: '2015-05-12 13:46:12',

author: „Thomas Lauria“.

context: '',

addInfo: ''

},{

source: „Das ist das Haus des Nikolaus“,

target: „This is the house of St. Nicholas“,

sourceLang: 'de', rfc5646

targetLang: 'en', rfc5646

matchRate: '100',

documentName: 'my file.sdlxliff',

DocumentShortName: 'shortnam.txt',

id: 'identifier',

type: 'Manual',

matchType: 'Exact',

segmentNumber: 123,

markupTable: 'XYZ',

timestamp: '2015-05-12 13:46:12',

author: „Thomas Lauria“.

context: '',

addInfo: ''

}]}

Errors:

- 400 Bad Request – if search, query or language parameters are missing or are not well formed.
- 404 Not Found – if TM file to given [TM_Name] in URL was not found
- 500 Server Error – for other technical problems.

http://opentm2/translationmemory/[TM_Name]/concordancesearch /?

POST – Performs a context search of the given search string in the proposals contained in a memory. Returns one proposal per request.

To OpenTM2:

{

searchString: 'Haus des Nikolaus',

searchType: 'source', // values can be source or target

searchPosition: 123// can be empty; Position where a search should start in the memory, see below

numResults: 1,

msSearchAfterNumResults: 100 //number of milliseconds the search will continue, after the first result is found. All additional results that are found in this additional time will also be returned until numResults is reached. If numResults is reached before msSearchAfterNumResults is reached, the search will abort. If msSearchAfterNumResults is reached before numResults is reached, search is also aborted. All found results are delivered in both cases.

[loggingThreshold:"2"]

}

From OpenTM2 HTTP 200 OK:

{

NewSearchPosition: '123:54', /returns NULL, if end of TM is reached, see below

results:[{

source: „Das ist das Haus des Nikolaus“,

target: „This is the house of St. Nicholas“,

sourceLang: 'de', rfc5646

targetLang: 'en', rfc5646

matchRate: '100',

documentName: 'my file.sdlxliff',

DocumentShortName: 'shortnam.txt',

id: 'identifier',

type: 'Manual',

matchType: 'Exact',

segmentNumber: 123,

markupTable: 'XYZ',

timestamp: '2015-05-12 13:46:12',

author: „Thomas Lauria“.

context: '',

addInfo: ''

},{

source: „Das ist das Haus des Nikolaus“,

target: „This is the house of St. Nicholas“,

sourceLang: 'de', rfc5646

targetLang: 'en', rfc5646

matchRate: '100',

documentName: 'my file.sdlxliff',

DocumentShortName: 'shortnam.txt',

id: 'identifier',

type: 'Manual',

matchType: 'Exact',

segmentNumber: 123,

markupTable: 'XYZ',

timestamp: '2015-05-12 13:46:12',

author: „Thomas Lauria“.

context: '',

addInfo: ''

}]}

Errors:

- 400 Bad Request – if search, query or language parameters are missing or are not well formed.
- 404 Not Found – if TM file to given [TM_Name] in URL was not found
- 500 Server Error – for other technical problems.

SearchPosition / NewSearchPosition

The NextSearchposition is an internal key of the memory for the next position on sequential access. Since it is an internal key, maintained and understanded by the underlying memory plug-in (for EqfMemoryPlugin is it the record number and the position in one record),
no assumptions should be made regarding the content. It is just a string which, should be send back to OpenTM2 on the next request, so that the search starts from there.
So is the implementation in Translate5: The first request to OpenTM2 contains SearchPosition with an empty string, OpenTM2 returns then a string in NewSearchPosition, which is just resend to OpenTM2 in the next request.