Specification of the OpenTM2 Rest-API V1.0

Overview

The OpenTM2 Rest-API provides the OpenTM2 memory related functions in form of a web server following the REST architecture. The OpenTM2 REST-API consist of three components:

1. Additional OpenTM2 API functions for the access to the memory system
2. A special markup table for the recognition of the Translate5 inline tags
3. A web service providing the REST-API.

Additional OpenTM2 API functions

New OpenTM2 API functions

1. EqfImportMemInInternalFormat
   Imports a memory using the individual memory files
   Input: MemoryName, NameOfInputFiles
   Output; return code, error message

2. EqfOpenMem
   Opens a memory.
   Input: Memory name
   Output: return code, error message, handle for the access to the memory

3. EqfCloseMem
   Close a memory opened using EqfOpenMem
   Input: MemoryHandle
   Output: return code, error message

4. EqfQueryMem
   Lookup the given segment in the memory
   Input: MemoryHandle, Segment with metadata
   Output: return code, error message, list of found matches with metadata

5. EqfSearchMem
   Search the given text in the memory
   Input: MemoryHandle, SearchString, SearchPosition
   Output: return code, error message, found match with metadata, new SearchPosition

6. EqfUpdateMem
   Update the memory with the given segment
   Input: MemoryHandle, Segment with metadata
   Output: return code, error message

Markup table for Translate5 inline tags

The markup table will be used to identify the Translate5 HTML tags in the segment data. The markup table will be packaged in form of an OpenTM2 markup table plugin.

REST server

Provided functions

1. Import TM in internal format (copy the memory files to the internal location), returns an error message when the memory exists
   Parameters: MemoryName, Files
2. Delete TM, close the memory when it is currently open, returns error message when the memory does not exist
   Parameters; MemoryName
3. Create new empty TM, returns an error message when the memory exists or when the language is not supported
   Parameters: SourceLanguage, MemoryName
4. Import TM in TMX format
   Parameters: MemoryName, TMX File
5. Open TM, the memory is opened for faster access, the server keeps an internal list of all opened memories
   Parameters: MemoryName
6. Close TM
   Parameters: MemoryName
7. Get matches from a TM
   Parameters: List of MemoryNames, text of source with Translate5 HTML tags, TargetLanguage, DocumentName, SegmentNumber
   Returns: List of found matches, each match includes the source text, the target text, the document name, the segment number, the author name, the timestamp, the type of the match (Manual, MT, GlobalMemory), the fuzziness of the match (value from 1 to 100)
8. Search TM for a given text (Concordance search)
   Parameters: MemoryName, Text being searched, SourceOrTarget flag, SearchPosition (if none given the search starts at the begin)
   Returns: found Match with the source text, the target text, the document name, the segment number, the author name, the timestamp, the type of the match (Manual, MT, GlobalMemory), the markup of the match, and the updated search position
9. Save new entry to TM (existing entry with same source and same context information gets updated and the timestamp of the entry is set to the current date/time)
   Parameters: MemoryName, sourceSeg (with Translate5 tags), targetSeg (with Translate5 tags), SourceLanguage, TargetLanguage, Author, DocumentName