DELIVERABLE

Project Acronym: iTranslate4

Grant Agreement number: 250405

Project Title: Internet Translators for all European Languages

---

1.1 Specification of common API for translation services

Revision: version 1.14

Authors:

Csaba Merényi and Kundráth Péter (MorphoLogic Számítástechnikai Kft., MOR)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

iTranslate4 Translation Service API

Nature: technical documentation

Table of contents:

1. Introduction

1.1. Purpose

This document describes the iTranslate4 Translation Service API. It is intended for the technical staff of project participants, as well as future partners wishing to join the iTranslate4 service. The technical documentation provided here defines the interface through which individual translation service providers can connect to the Central Server Application of the iTranslate4 on-line translation service. Each consortium member as well as future partners needs to implement this interface.

1.2. Scope

This document covers the specification of the interface between the Central Server Application and individual translation service providers only. Other APIs are used in the iTranslate4 project, which are described in different documents.

1.3. Definitions, acronyms and abbreviations

The following terms and abbreviations are used in this document:

- Central Server Application (CSA): a software component in the iTranslate4 architecture. The CSA runs on a central server machine that forms part of the core iTranslate4 infrastructure and it is responsible for passing translation requests from the users to translation services, ranking alternative translations, handling user feedback, logging, etc.
- Translation Service (TS): any one of the individual translation services provided by consortium members or future partners. TSs are hosted either by remote servers maintained by service providers, or locally on dedicated TS servers. In both cases TSs connect to the CSA as web services through the API specified here.

1.4. References

The present document makes reference to the following technical documents: N/A

1.5. Overview

The rest of this document is structured as follows.

Section 2 (API Specification) describes the Translation Service API in technical terms. In 2.1 (General Information about the Interface) some basic principles are laid down. 2.2 (iTranslate4 API functions) lists all API functions and describes them in full detail.

Section 3 (Examples) shows examples of API calls and expected responses.

2. API Specification

2.1. General Information about the Interface

2.1.1. Request format

Function calls are sent by the CSA to the Translation Services either as HTTP POST or HTTP GET requests. A special POST will be used in standard operation, while normal POST and GET must be implemented for testing and debugging purposes. The generic request format is as follows:

Direct POST:

```
http://server:port/...?func=<function_name>
content: <object>
```

POST:

```
http://server:port/...
content: func=<function_name>&data=<object>
```

GET:

```
http://server:port/...?func=<function_name>&data=<object>
```

The *func* parameter is used to pass the function name (e.g. *translate*, which executes a translation.) The list of valid <function_name>s is described in 2.2. below.
In the case of normal POST and GET requests the *data* parameter is used to pass a JSON object that contains function-specific options and content. The same object is sent as *content* in POST requests. Valid <object> formats are described in 2.2.x for each function.

2.1.2. Response format

The response is a JSON object. Valid replies are specified for each function in 2.2.x

2.1.3. Character encoding

All data sent and received through the iTranslate4 Translation Service API is UTF-8 encoded. For debugging purposes handling of the standard JSON encoding for characters is required. The format of this is \uXXXX where XXXX is the hexadecimal code for that character in the Unicode table.

2.1.4. Error handling

Standard HTTP error codes are used to indicate problems that are not specific to a particular TS or any given TS function. Error codes are defined in 2.3. If a TS is running normally but finds its input invalid in any way or needs to return error information on any particular query for any reason, the JSON object sent in the reply may contain an “err” field. See 2.3. for more information.

2.2. iTranslate4 API functions

Valid <function name>s, their purpose and associated data formats are listed below. TSs are not required to implement all of these functions. Whether implementing a given function is obligatory

or not will be indicated by the words REQUIRED and OPTIONAL, respectively. Fields in the JSON data objects may also be optional, which will be indicated the same way.

2.2.1. The *info* function

purpose: The *info* function returns all information on a translation service needed by the CSA (and the web server) at initialization. Being able to get this information dynamically allows the iTranslate4 infrastructure to be maintained and extended with minimum effort.
status: REQUIRED
call format (GET): http://server:port/...?func=info
response data format:

info_obj

version	REQUIRED	STRING
vendor	REQUIRED	STRING
engine	REQUIRED	STRING
logo	REQUIRED	STRING
lp	REQUIRED	ARRAY of [STRING, STRING]
features	REQUIRED	*feat_obj*

version: the current version of the translation service. It cannot exceed 16 characters and may not contain white spaces.
vendor: name of the service provider as it appears on the user interface of the iTanslate4 website
engine: name of the translation engine as it appears on the user interface of the iTanslate4 website
logo: the URL of an image file containing the current logo of the service provider. Size and format requirements: width: 100px; height: 25px; image type: gif
lp: returns the list of supported language pairs. Each element in the array is a pair of IANA language codes in the form of [<source language>,<target language>].
features: returns the list of supported features as a *feat_obj* object. All members of *feat_obj* are arrays of pairs of STRINGS, where each element stands for a source language – target language pair (IANA language codes) that supports the given feature. Instead of language codes a single "*" character may be used to represent all languages.

feat_obj

wid	OPTIONAL	ARRAY of [STRING,STRING]
unkn	OPTIONAL	ARRAY of [STRING,STRING]
segm	OPTIONAL	ARRAY of [STRING,STRING]

alt	OPTIONAL	ARRAY of [STRING,STRING]
syn	OPTIONAL	ARRAY of [STRING,STRING]
html	OPTIONAL	ARRAY of [STRING,STRING]

wid: support for word alignment, for more info on wid see 2.2.2
unkn: support for marking of unknown words
segm: support for sentence segmentation
alt: support for alternative translations
syn: support for synonyms/alternative translations of words
html support for the html translation via "doctranslate" (see 2.2.3)

2.2.2. The *translate* function

purpose: The translate function allows the CSA to send a block of text to the TS for translation using the specified linguistic settings.
status: REQUIRED
call format (GET): http://server:port/...?func=translate&data=<tr_obj>
request/response data format:
Please note that for the *translate* function the response data format is the same as the request data format, which allows the CSA to pass the result of a translation request to another TS without significant transformations in the case of linked translations. Data fields are qualified as IN and/or OUT (default is both), but the presence of any data field in either input or output must be tolerated by the TS as well as the CSA.

tr_obj

src	REQUIRED IN	STRING
trg	REQUIRED IN	STRING
dom	OPTIONAL IN	STRING
sgms	REQUIRED	ARRAY of *segm_elem*
uid	OPTIONAL IN	STRING
tid	OPTIONAL IN	STRING

src: source language specified as IANA language tag
trg: target language specified as IANA language tag
dom: (domain) Specifies the topic of the text to be translated. TSs may use specialized dictionaries depending on this setting. The list of allowed values and their meaning is defined in the following table:

general	default behavior / no specialized dicitionaries

art	arts & humanities and social sciences (art and culture, architecture, music and dance, literature and theatre, visual arts, humanities, education, history, linguistics, philosophy, psychology, work and social life, society and welfare, religion, politics, university, etc.)
law	law & administration (customs, criminality, quality assurance etc.)
sci	natural science (mathematics, physics, chemistry, geography)
tech	technology (engineering, construction, electrical engineering, power generation, environment, mechanical engineering, exploitation of raw materials, optics and photography, manufacturing, wood processing, productive industry, cars etc)
info	information technology (informatics, computers, data processing etc.)
tel	telecommunication and media (tele industry, telephony, etc.)
bio	health and biology (pharmaceuticals, biology, food and nutrition science)
eco	economy and trade (finance, business, logistics etc.)
agr	agriculture
travel	transport and travelling (traffic, rail etc.)
mil	military (armament, naval, maritime etc.)
sport	sports & games (leisure activities, cooking etc.)

sgms: (segments) represents the text to be translated. Each *segm_elem* object in the array represents one paragraph.

uid: a unique ID identifying logged-in users. Unique user IDs allow TSs to handle user dictionaries or apply other user-specific settings.

tid: (translation ID) a unique ID sent by the CSA to identify each request. This ID may be used by TSs to log their activity. The *tid* need not be returned in the output tr_obj.

segm_elem

Units	REQUIRED	ARRAY of *unit_elem*

units: contains a sequence of translation units. If the TS supports segmentation (according to the results of the info function), then the CSA does not perform sentence segmentation, and the input object has all sentences of the paragraph in the same *unit_elem*. If the TS does not support segmentation, or (in case of linked translations) this is not the first TS to call, then the input object will be already segmented. Segmentation is indicated by the align parameter in each unit, and ***if align parameter***

is found sentence-level segmentation is not allowed. If segmentation is required (and the TS supports sentence-level segmentation) then the output should contain each translated sentence in a separate *unit_elem* object. It is assumed that a sentence may have alternative translations, which henceforth will be referred to as v*ariants.* Variants should be returned as separate *unit_elem* objects. The fact that a given *unit_elem* represents a variant is not shown explicitly, but variants have the same values in their *align* fileds. (see the definition of *align* below).

unit_elem

text	REQUIRED	STRING
unkn	OPTIONAL	ARRAY of <word pointer>
wid	OPTIONAL	ARRAY of ARRAY of STRING
align	OPTIONAL	[<word pointer 1>,<word pointer 2>]
syn	OPTIONAL	ARRAY of [<word pointer 1>,<word pointer 2>, STRING]

The above definition of *unit_elem* refers to *word pointers.* A *word pointer* is a zero-based index addressing a word position in the *text* field, so '0' refers to the first word of *text*, '1' refers to the second word, etc. When determining word positions, only whitespace characters are considered to be word boundaries.

text: one translation unit (sentence, or sequence of unsegmented sentences) represented as a string. In the input *tr_obj* this field may contain several sentences. In the output *tr_obj,* (after segmentation by the TS) each *unit_elem* should contain exactly one sentence.

unkn: (unknown words) Indicates unknown words in the translation as an array of *word pointers.*

wid: (word ID) In the input *tr_obj* the CSA assigns each word a unique *word ID.* The containing array lists the word IDs for each whitespace-separated word of the *text* field as single-element arrays of STRING. (e.g. [[0],[1],[2]]) In the output *tr_obj* the *wid* field indicates which output word corresponds to which input word. The length of the array must be equal to the number of words in *text*. Each position in the array represents a word at the same position in *text*, while the value of the array element is an array listing the word IDs of the source words that correspond to the given output word, or empty if no such correspondence exits. This information, if present, may be used to keep the formatting of the source text at the word level.

align: this field provides sentence-level alignment information. *align* is a pair of word pointers, where <word pointer 1> and <word pointer 2> identify the beginning and the end of the sentence in the input *text* field. NOTE: alignment info is needed because sentence segmentation is done by the TS, so the input object stores all sentences in a single *unit_elem,* while the output may store the sentences in an array of *unit_elem* objects. Also note that the same *align* values in different *unit_elems* indicate variants

(i.e. alternative translations of the same source sentence). If this field is present in the input then either leave it unchanged or remove it from the output.

syn: (synonym) represents alternative translations at the word level. An element in the *syn* array is a [<word pointer 1>, <word pointer 2>, <string>] triplet, where < word pointer 1> and <word pointer 2> identify the beginning and the end of a range in the output *text*, field and the <string> contains the alternative translation of that sequence of words. Multiple synonyms can be represented by adding to the array more than one [<word pointer 1>, <word pointer 2>, <string>] triplet pointing to the same range.

2.2.3. The *doctranslate* function

purpose: The *doctranslate* function may be implemented by vendors who wish to utilize their document translation capabilities as a single module rather than rely on the CSA for parsing incoming documents and sending their contents as separate blocks of text to them through the standard *translate* function defined above. *Doctranslate* allows the CSA to send a complete document or a link to the TS for translation using the specified linguistic settings.
status: OPTIONAL
call format (GET): http://server:port/...?func=doctranslate&data=<tr1_obj>
request data format:

tr1_obj		
src	REQUIRED	STRING
trg	REQUIRED	STRING
dom	OPTIONAL	STRING
type	REQUIRED	STRING
ext	OPTIONAL	STRING
dat	OPTIONAL	STRING
uid	REQUIRED	STRING
tid	REQUIRED	STRING

src: The IANA language code of the source.
trg: The IANA language code of the target.
dom: Domain for the translation. For possible values see the table above in 2.2.2
type: The format of the data to be translated. Current supported values: **html**
ext: The url of the data to be translated. REQUIRED if *dat* is empty.
If *dat* is not given, then the data to be translated has to be downloaded from this location. If *dat* is not empty then it already contains this data.
dat: The data to be translated. This may contain binary data (doc or pdf files, if these are supported – they are not supported now)
uid: a unique ID identifying logged-in users. Unique user IDs allow TSs to handle user dictionaries or apply other user-specific settings.
tid: (translation ID) a unique ID sent by the CSA to identify each request. This ID may be used by TSs to log their activity. The *tid* need not be returned in the output tr_obj.

response data format: the translated document.

2.3. Error codes

General error conditions related to the operation of TSs shall be reported as HTTP error codes.

400 – Bad Request (for example invalid character code)
403 – Forbidden (for example invalid CRC (if implemented))
408 – Request timeout (translation could not be finished within given time interval)
500 – Internal Server Error (translator experiencing temporary technical difficulties)
503 – Service Unavailable (translator unavailable)

Errors related to the queries sent to TSs may be indicated in the JSON object sent as a reply. All replies may contain an "err" field at the top level in the structure. The "err" field should contain a string in the format "ERR_CODE ERR_STRING", where ERR_CODE is a numeric error code and ERR_STRING is a simple description of the problem in English. When creating ERR_STRINGs, please keep in mind that these error messages may be displayed to users.

A common set of ERR_CODEs will be defined soon …TBD

3. Examples

3.1. translate request and response

REQUEST format:

```
http://....?func=translate&data=<object>
```

the examples below demonstrate a linked translation from German to French through English.

REQUEST object (CSA -> TS1):

```
{
  src: 'de',
  trg: 'en',
  dom: '',
  sgms:
  [
    {
      units:
      [
        {
          text:'Andrew hat seine Aufgabe erledigt. Er kann jetzt nach Hause gehen.',
          wid: [[0],[1],[2],[3],[4],[5],[6],[7],[8],[9],[10]]
        }
      ]
    }
  ],
  uid: 0,
  tid: 1276601710.4567
}
```

RESPONSE object (TS1 -> CSA):

```
{
  sgms:
  [
    {
      units:
      [
        { text:'Andrew has done his job.',
          wid:[[0],[1],[4],[2],[3]],
```

```
          align:[0,4],
          syn:[[3,3,'her']]
        }
        { text:'Now he can go home.',
          wid:[[7],[5],[6],[10],[9]],
          align:[5,10]
        }
      ]
    }
  ],
}
```

REQUEST object (CSA -> TS2):

```
{
  src: 'en',
  trg: 'fr',
  dom: '',
  sgms:
  [
    {
      units:
      [
        { text:'Andrew has done his job.',
          wid:[[0],[1],[4],[2],[3]],
          align:[0,4],
          syn:[[3,3,'her']]
        }
        { text:'Now he can go home.',
          wid:[[7],[5],[6],[10],[9]],
          align:[5,10]
        }
      ]
    }
  ],
  uid: 0,
  tid: 1276601710.4567
}
```

RESPONSE object (TS2 -> CSA):

```
{
  sgms:
  [
    {
      units:
      [
        { text:'Andrew a réalisé son travail.',
          wid:[[0],[1],[4],[2],[3]],
          align:[0,4],
          syn:[[3,3,'her']]
        }
        { text:'Maintenant il peut rentrer à la maison.',
          wid:[[7],[5],[6],[10],[9],[9],[9]],
          align:[5,10]
        }
      ]
    }
  ]
}
```

Location of the common API:

http://itranslate4.eu/project/deliverables.html

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Author	Organisation	Description
V1.14	27/2/2011	Kundráth Péter	MOR	description of align and syn fields in units fixed (2.2.2)
V1.13	14/2/2011	Kundráth Péter, Merényi Csaba	MOR	clarified API calls (2.1.1), extended character encoding requirements (2.1.3), added html feature (2.2.1), changed units description (2.2.2)
V1.12	1/7/2011	Kundráth Péter, Merényi Csaba	MOR	error handling – "err" field added
V1.11	10/18/2010	Kundráth Péter	MOR	doctranslate removed: trs and sess, added uid and tid
v1.10	10/11/2010	Kundráth Péter	MOR	feature field changed – now includes list of language pairs that support a given feature
v1.9,	9/29/2010	Kundráth Péter	MOR	feature field is introduced into info function, *name* field changed to *vendor* and *engine* in info function
v1.8	9/20/2010	Kundráth Péter, Merényi Csaba	MOR	doctranslate function added
v1.7	9/1/2010	Kundráth Péter, Merényi Csaba	MOR	service information function renamed
V 1.6	7/9/2010	Kundráth Péter, Merényi Csaba	MOR	representation of variants changed, now *units* is simply an array, variants are stored in this array without any hierarchy; *wid* changed to array of array to allow many-to-many mapping; examples corrected/updated.
V1.5	6/25/2010	Kundráth Péter, Merényi Csaba	MOR	more examples, syn corrected, user dictionary removed
V1.4	6/21/2010	Kundráth Péter, Merényi Csaba	MOR	data format changed – added wid, align, removed typo; added GetServiceInfo function

V1.3	6/7/2010	Kundráth Péter, Merényi Csaba	MOR	representation of sentence variants changed, representation of text formatting changed
V1.2	6/4/2010	Kundráth Péter, Merényi Csaba	MOR	use of HTTP POST/GET clarified
V1.1	6/3/2010	Kundráth Péter, Merényi Csaba	MOR	

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.