# #Introduction

The translation industry is affected by a lack of interoperability due to the lack of open standards and the inveterate practice of tool manufacturers to manipulate existing standards to lock their customers in.

This document is the result of the efforts of volunteers of the WG1 for the GALA TAPICC initiative to provide a few recommendations for developing general open and ﬂexible web-service APIs that would address the most common use cases in translation and localization project management.

Rather than yielding to the audacity of describing a universal API in its objects, methods, and calls (possibly providing some code snippets), the TAPICC WG1 members agreed to issue a set of recommendations to manufacturers of translation tools and platforms for developing their own APIs to perform the basic tasks in a translation assignment.

We hope that a set of recommendations will be welcome and followed, and in any case better received than an allegedly universal API that might be perceived as an over-standardization effort.

So far, most of the eﬀorts in deﬁning standards for translation processes have focused on ﬁle formats. This set of recommendations addresses system interactions.

We believe that every CMS could embrace these recommendations and easily implement a TAPICC-compliant API, and once enough CMSs have adopted it, TMSs will follow. Nevertheless, most organizations have some sort of acquired/developed process for billing, quoting etc. in place, so these recommendations will address what metadata should necessarily be transferred without forcing any substitution.

In summary, this document describes all the major entities for an API to consider when communicating with a system and/or helping information exchange between systems.

# #Prerequisites

A TAPICC-compliant API should operate on three consecutive integration “levels” with different functionalities.

The first level (L1) shall deal with the exchange format, the one used to exchange data between systems and for the package structure of the payload.

The second level (L2) shall deal with the mechanism of exporting/importing the payload. Only one exchange folder shall be used instead of a predefined and rigid folder structure. The TMS should host it. CMS and PIM should consume it. File naming conventions will be specified by WG2.

The third level (L3) shall deal with the communication mechanism.

Each component of the TAPICC infrastructure shall be open, to help development, testing, and implementation.

The transfer mechanism shall be protocol-independent and based on the REST architectural style, with units being transmitted including both header information and the actual data being sent. JSON shall be used as the file format to transmit data objects. More specifically, a TAPICC-compliant API shall exploit separate payload description and encapsulation from transactions, using XML for the first and JSON for the latter, with prepared material in XLIFF format and standardized package wrapper.

OAuth 2.0 and XACML shall be used for authentication purposes, possibly supported by a 2FA mechanism.

# #Goals and purposes

Based on above prerequisites, TAPICC-compliant APIs are expected to:

1. Expedite transactions;
2. Help integration;
3. Help interoperability;
4. Reduce intermediation.

Any TAPICC-compliant API shall then be capable of:

1. Being implemented across all systems;
2. Being consumed from any point similarly;
3. Conveying bids and quotes;
4. Submitting raw or prepared material;
5. Getting translated material back.

Any TAPICC-compliant API shall also cover system-to-system integration in such a way as to:

1. Exchange project data between CMS and TMS;
2. Exchange payloads between CMS and TMS;
3. Provide for continuous storage of translation project data to be available separately for QMS;
4. Exchange payloads between TMS and translation tools.

# #Topics

Rather than trying to anticipate every possible use case, thus forcing developers into a considerable eﬀort to implement the ensuing sophistication, this document provides a set of guidelines to help designers and developers to prototype new TAPICC-compliant APIs, even when they have little or no knowledge about translation services.

The major benefit expected by TAPICC-compliant APIs will be a general simplification in the interaction of translation resources and services together with the fractioning of complex tasks into several small and simple requests.

## ##Interactions

This section lists the basic set of features to be found in a TAPICC-compliant API, as a set of recommendations for organizations that may be interested in developing their own APIs.

The most basic feature is the request of a translation and the delivery of results.

With many other possible interactions are also described here to help software developers build an API that could be rapidly and easily implemented, possibly without assistance.

The list is laid out with a title summarizing the interaction and a description outlining the operation to be performed.

### ###Essential interactions (CMS<>TMS)

Interaction: \*\***CMS Login**\*\*

Description: Login on the CMS.

Interaction: \*\***Request of a translation**\*\*

Description: The client requests the translation via a POST request. The translation request shall contain the parameters of the request.

Interaction: \*\***Project creation**\*\*

Description:

Interaction: \*\***Project info**\*\*

Description: CMS sends metadata to the TMS.

Interaction: \*\***Set AQL**\*\*

Description: The client sets the desired quality level for the translation from a list of available parameters.

Interaction: \*\***Check status**\*\*

Description:

Interaction: \*\***Cancel project**\*\*

Description:

Interaction: \*\***Getting the translation**\*\*

Description: Once the text is translated, the status of the request is updated and the resource is available via a GET request.

Interaction: \*\***CMS** **Upload**\*\*

Description: Upload of payload on the CMS.

Interaction: \*\***Project updates**\*\*

Description: TMS pushes updates for translation projects to CMS.

Interaction: \*\***Status** **reporting**\*\*

Description: TMS pushes status report to CMS.

Interaction: \*\***Request a review**\*\*

Description:

Interaction: \*\***Accept/reject project/task**\*\*

Description:

Interaction: \*\***CMS Logout**\*\*

Description: Logout from the CMS.

Interaction: \*\***TMS Login**\*\*

Description: Login on the TMS.

Interaction: \*\***Project creation**\*\*

Description: The LSP Initializes a new project and adds attributes. A project may also consist of a single segment.

Interaction: \*\***Set AQL**\*\*

Description: The LSP accept/reject/amend client’s AQL.

Interaction: \*\***Check status**\*\*

Description:

Interaction: \*\***Cancel project**\*\*

Description:

Interaction: \*\***Update project**\*\*

Description: The LSP updates an already submitted project. Metadata may be updated too.

Interaction: \*\***Accept/reject project/task**\*\*

Description:

Interaction: \*\***Submit translations back to the requester** \*\*

Description: The LSP submits a translation to CMS and terminates project.

Interaction: \*\***TMS Logout**\*\*

Description: Logout from the TMS.

### ###Ancillary interactions (TMS<>translation tool)

Interaction: \*\***Forward a request to a translator or an automated service**\*\*

Description:

Interaction: \*\***TMS** **Upload**\*\*

Description: Translators upload payload on the TMS. The resource created shall be uniquely identiﬁed, and shall be retrieved via a standard GETrequest.

Interaction: \*\***Find project**\*\*

Description:

Interaction: \*\***Get a translation back from a translator or an automated service**\*\*

Description:

Interaction: \*\***Request a review**\*\*

Description:

Interaction: \*\***Request status** **report**\*\*

Description: