

| Logo  Description automatically generated |
| --- |
|  |
| **Recommendation report on connecting e-services to the interoperability infrastructure**  **Kilian KLEIN (EUF)**  14/04/2023 |
|  |

**Table of Contents**

[**1. Introduction 0**](#_29333l8s9sxl)

[1.1. Definition 0](#_z5m85dusz5sq)

[1.2. How to read this document 0](#_ka4h4cs4gbb2)

[**2. The eSignature CEF building block 1**](#_87nrkwuzbf1x)

[2.1. Compatibility with eIDAS 1](#_q5owtxsvpzr4)

[2.2. Authentication via MyAcademicID 1](#_jotam1rxf8fb)

[2.3. Integration in the OLA platform 1](#_q2qib2hrdhcn)

[2.4. Use of certification authority (CA) 1](#_jycd1cvvxw4z)

[2.5. Other tools/systems in the interoperability infrastructure that would benefit from the eSignature integration 1](#_tg7pbu4u2o5c)

[2.6. Recommendations on eSignature CEF building block usage 1](#_t5ty68gjzlo2)

[**3. The eArchive CEF building block 2**](#_edg1kgq2h78g)

[3.1. Integration in the OLA platform 2](#_cnqcrg74lczj)

[3.2. Compatibility with eIDAS 2](#_dhm6zdccsf1n)

[3.3. Other tools/systems in the interoperability infrastructure that would benefit from the eArchive CEF building block integration 2](#_x5zl7drswqg7)

[3.4. Recommendations on eArchive CEF building block usage 2](#_6ybo63ao6lpu)

[**4. The eTranslate CEF building block 3**](#_gdyvhxu7cxfe)

[4.1. Developed proof-of-concept: Drupal CMS 3](#_t9z4rm9hvmqj)

[4.2. Tools/systems in the interoperability infrastructure that would benefit from the eTranslate CEF building block integration 3](#_h8shecjjxf9l)

[4.3. Recommendations on eTranslate CEF building block usage 4](#_7on9ywk2ud9v)

[4.3.1. Use case: Translate documents / texts 4](#_47df9hiod5st)

[4.3.2. Use case: Translate a website 5](#_eokq2tdbdnn7)

[4.3.3. Use case: Translate data templates 6](#_dblhi9uutw4g)

[**5. The student housing web application proof-of-concept 7**](#_3p071jko209w)

[5.1. Compatibility with the Erasmus+ App 7](#_iho5ins4cnhi)

[5.2. Log in using academic credentials via MyAcademicId 7](#_6ofj05yq0k3)

[5.3. Sign contract using eSignature 7](#_e66bor3j1j8k)

[5.4. Recommendations on usage of the student housing web application proof-of-concept 8](#_x951h4l04lps)

[To be completed after reading specs / documentation of the housing PoC 8](#_exz6b2ln9jnk)

[**6. Activity 1’s eCard app 9**](#_hxo6kk9jnx6l)

[6.1. Compatibility with eIDAS 9](#_3a2k63uo337o)

[6.2. Recommendations on eCard app usage 9](#_c1xexpd35nph)

# **Introduction**

## Definition

The term “**interoperability infrastructure**” must be understood as the broad notion of “every digital infrastructure or tool related to the management of student mobility”.

For example, this definition matches:

* the *EWP network,*
* the *OLA platform*,
* the *Erasmus+ Dashboard,*
* *MyAcademicID*
* and the *Erasmus+ App*.

## How to read this document

This document presents how the different e-services, developed or analyzed during the EDSSI L2 project, have used the interoperability infrastructure and offers recommendations on how other parts of the interoperability could use those e-services.

This document is based on the findings and deliverables of other tasks and activities from the EDSSI L2 project and as such, will only provide brief explanations of what each e-service is and how they function. For more detailed information on each e-service, please refer to their respective documentation.

# 

# **The eSignature CEF building block**

## 

*Based on Task 2.1’s “*[*Updated requirements analysis report*](https://docs.google.com/document/d/1KgJKCp2p7pmW3q9_hYzgNhW5bRppFeI4/edit?usp=sharing&ouid=106005105817912639512&rtpof=true&sd=true)*” and … .*

To be completed as the reports for eSignature service is not available in the drive yet and no mention of the actual integration of the OLA is available yet.

### Compatibility with eIDAS

To be completed…

### Authentication via MyAcademicID

To be completed…

### Integration in the OLA platform

To be completed…

### Use of certification authority (CA)

To be completed…

### Other tools/systems in the interoperability infrastructure that would benefit from the eSignature integration

To be completed…

### Recommendations on eSignature CEF building block usage

To be completed…

# **The eArchive CEF building block**

*Based on Task 2.1’s “*[*Updated requirements analysis report*](https://docs.google.com/document/d/1KgJKCp2p7pmW3q9_hYzgNhW5bRppFeI4/edit?usp=sharing&ouid=106005105817912639512&rtpof=true&sd=true)*” and … .*

To be completed as the reports for eArchive service is not available in the drive yet and no mention of the actual integration of the OLA is available yet.

### Integration in the OLA platform

To be completed as this integration is mentioned in the [*Updated requirements analysis report*](https://docs.google.com/document/d/1KgJKCp2p7pmW3q9_hYzgNhW5bRppFeI4/edit?usp=sharing&ouid=106005105817912639512&rtpof=true&sd=true) but as far as I remember this idea was sort of abandoned or put on hold.

### Compatibility with eIDAS

To be completed…

### Other tools/systems in the interoperability infrastructure that would benefit from the eArchive CEF building block integration

To be completed…

### Recommendations on eArchive CEF building block usage

To be completed…

# **The eTranslate CEF building block**

*Based on Task 2.4’s “eTranslate CEF building block usability report”.*

The *eTranslate CEF building block* uses a machine-learning algorithm that focuses on translating sentences, or segments of sentences, on the theme of higher education.

It has already been trained on more than a billion exact segment translations and is constantly subjected to more of them in order to improve its results.

The eTranslate CEF building block computes translations asynchronously. This means that each translation request will not instantly receive the translation in response but will rather enter a queue that will be addressed when the computing capacity is available and then answered when the translation is complete.

The building block offers two, non-exclusive, methods of receiving the translated content:

* *By email* or
* *Machine-to-machine* calling back an endpoint URL defined at the request of the translation.

## Developed proof-of-concept: Drupal CMS

As mentioned in the [eTranslate CEF building block usability report](https://docs.google.com/document/d/11-9ihaLYDAAglclofc2V4R5zXcb6janP/edit?usp=sharing&ouid=106005105817912639512&rtpof=true&sd=true): “*There is a sample scenario on how to use the eTranslation web service through the Drupal CMS*”.

This proof-of-concept, developed in EDSSI L2, demonstrates the usage of the *eTranslate CEF building block* within a CMS application using the “*by email*” method for receiving the translation. The machine-to-machine feature is yet to be demonstrated in a proof-of-concept.

Using the Drupal CMS technologie was no coincidence as several of the key pieces of the interoperability infrastructure are also based on the Drupal CMS such as the OLA platform, and the Erasmus+ App.

## Tools/systems in the interoperability infrastructure that would benefit from the eTranslate CEF building block integration

The [eTranslate CEF building block usability report](https://docs.google.com/document/d/11-9ihaLYDAAglclofc2V4R5zXcb6janP/edit?usp=sharing&ouid=106005105817912639512&rtpof=true&sd=true) refers to many pieces of the interoperability infrastructure that would benefit from using the eSignature CEF building block:

* Course catalogs
* The Erasmus+ Dashboard’s website,
* The OLA’s (Online Learning Agreement) website,
* The EWP wiki,
* Learning agreements (LA) templates,
* Transcript of records (ToR) templates,
* Inter institutional Agreements (IIA) templates and data (factsheet),
* MyAcademicID’s authentication interfaces,
* The EWP registry,
* International Relations Offices’ websites,
* Student Service Providers (SSP) websites and data,
* Etc.

## Recommendations on eTranslate CEF building block usage

The eTranslate CEF building block can already be used and trusted for high accuracy for the translation of text on the theme of higher education. The machine learning algorithm it uses has already been trained to translate more than a billion segments of sentences and will become more and more powerful with time as it gets trained with more segments.

The asynchronous nature of this CEF building block’s translations functionality requires the requester to adapt their processes as “live” translations are not possible.

The following translation use cases relevant to higher education will list the different steps to follow when using the eTranslate building block.

### Use case: Translate documents / texts

Translating documents or texts can happen in different contexts:

* One-time quick translation request
* Addition of an “end-user” translation functionality to an application
* Automated translation

For **one-time translation requests**, it is recommended for the requester to visit the official tool available at <https://webgate.ec.europa.eu/cas/login>and to request their translation directly on the form it offers.

In order to **add translation functionality for the end-user to an application**, the user interface must clearly explain to the end-user what the asynchronous aspect of the translation functionality implies. The end-user will receive their requested translation “Only when it is ready” as opposed to receiving after actively waiting upon request like they would usually expect from an online form/functionality.

Moreover, concerns about data privacy must be addressed and transparently exposed to the end-user:

* Storing the original request text and/or the asynchronously received translated text implies potentially storing private data on the requesting application
* The eTranslate CEF building block will also store the request and translation temporarily for which the end-user must be warned as the building block then acts as a third party.

Must be noted that using only the “email” method of receiving the translation reduces the GDPR impact but does not entirely remove it.

Similarly, **automated translation capacities** added to a digitalized system have to cover the same GDPR concerns.

Explaining the asynchronous aspect of the translation system in this automated machine-to-machine use case is not as critical as when allowing the end-user to request for translations since this use case will not raise expectations of instant results from the end-user.

Basic flow for such automated translations:

1. Upon creation/update of the resource to be translated: send the translation request to the eTranslate web service with the URL pointing to the functionality of handling the reception of the translation
2. The eTranslate CEF building block, when available, computes the requested translation
3. The eTranslate web service calls the given URL back to send the computed translation
4. The requesting application processes the computed translation, saves it and renders it available to the end-user

### Use case: Translate a website

The translation of a website must be separated in 2 different concepts that are handled differently and at different moments because of their nature:

* The **website’s literals** (text belonging to the website itself, such as menu items, button labels, cookie policies, etc.) which are constant between 2 source code deployments
* The **website’s dynamic content** (articles, comments, resources added by the end-users) which can change at any moment.

The **translation of a website’s literals** is not strongly affected by the asynchronous nature of the eTranslate web service since changes in the source code happen before deployment. It is however recommended to request the translation of those literals a few days before the predicted deployment date to give time to the CEF building block to compute the requested translations.

Although the eTranslate web service knows how to handle the HTML format, in most of the relevant cases, the website to translate will be using dynamic languages to computes their HTML (such as PHP or ReactJS) rendering it impossible for the eTranslate web service to compute. A common best practice for multi-language websites is to only refer to translation keys in place of a website’s literals and to use a translation file, per desired language, exposing the respective translation for each of the literals keys.

Using this best practice, the basic flow for translation of the website’s literals is the following:

1. The application requesting translation formats the original language’s translation file’s contents to be understood by the eTranslate CEF building block (by removing the different translation keys that are potentially not human-readable)
2. The application sends request for translation to the eTranslate web service
3. The eTranslate CEF building block, when available, computes the requested translation
4. The eTranslate web service calls the given URL back to send the computed translation
5. The requesting application processes the computed translation and saves it in the format of the desired language’s translation file with the matching translation keys

The **automated translation of the website’s dynamic content** follows the basic flow presented in the “Translate documents / texts” section.

### Use case: Translate data templates

This use case is the most straightforward since data templates such as Learning agreements and Transcript of records use very well known and constant vocabulary.

Because of this, even if the *eTranslate CEF building block* would be used manually in order to translate those templates, the manual labor would be minimal and the translation results would be expected to be very close to perfection.

# **The student housing web application proof-of-concept**

The task “MS2.5 Housing web application proof-of-concept” aims to develop a proof-of-concept of a web-based application managing use cases related to housing for international students.

This proof-of-concept’s main use cases were scoped to the following:

* Log in to the application with academic credentials
* Search through a list of accommodations with potential filtering/sorting
* **Generate a rental contract based on accommodation offer**
* **Sign generated contract using e-signature**

The two latter use cases are the outputs with the most value for this housing web application proof-of-concept.

## Compatibility with the Erasmus+ App

**The Erasmus+ App and the HOME project**

The Erasmus+ App’s purpose is to become the central point for digital functionalities related to students' international mobilities. It includes its own “housing search tool” developed during the *HOME project*. Based on the HOME research, this project delivered:

* A list of normalized quality labels for international student housing offers calculable using the fields present in the DDS,
* a **Digital Data Standard** representing the format, including both required and optional fields, to describe a housing offer for an international student,
* a set of digital tools based on the results of such research.

The Erasmus+ App’s “housing search tool” shares many common requirements with the *student housing web application proof-of-concept* developed within the EDSSI L2 project. Therefore, efforts have been set in place to enhance compatibility and cooperation between the 2 projects.

**Representing offers using the Erasmus+ App’s HOME Digital Data Standard**

The HOME Digital Data Standard (DDS hereafter) aims to provide a common format for the representation of housing offers within the whole interoperability infrastructure. With a common format, compatibility between systems is automatically ensured.

Within the implementation of the *housing web application proof-of-concept*, the use of the DDS to represent housing offers enables two very interesting possible implementations in the future:

* Possibility to use the HOME API within the *housing web application proof-of-concept* to recover and expose housing data (should the legal framework allow for it),
* Addition of the rental contract generator with e-signature in the *Erasmus+ App*.

## Log in using academic credentials via MyAcademicId

To be completed after reading specs / documentation of the housing PoC

## Sign contract using eSignature

To be completed after reading specs / documentation of the housing PoC

## Recommendations on usage of the student housing web application proof-of-concept

### To be completed after reading specs / documentation of the housing PoC

# **Activity 1’s eCard app**

To be completed… Should we even have this in the report?

## Compatibility with eIDAS

To be completed…

Add something about Activity 1 ecard (the ecard app, if ever put in production, and if the SSP modules is fully functional, should figure on the SSP lists of each university if they allow/implement it)

## Recommendations on eCard app usage

To be completed…

### 

| Logo  Description automatically generated  [**www.edssi.eu**](https://edssi.eu/) |
| --- |