Grant Agreement 297292

EUROPEANA INSIDE

Technical Specification

Deliverable number	D2.5
Dissemination level	Public
Delivery date	November 2012
Status	V1.0 Final
Author(s)	K-INT

This project is funded under the
ICT Policy Support Programme part of the
Competitiveness and Innovation Framework Programme.

Revision History

Revisio n	Date	Author	Organisation	Description
V0.1	2012-10-31	Neil Smith	K-INT	Document structure
V0.2	2012-11-09	Rich Bruin; Rob Tice; Gill Osguthorpe; Ian Ibbotson	K-INT	Initial version of architecture and component requirements sections
V0.3	2012-11-16	Rich Bruin; Neil Smith; Rob Tice	K-INT	Initial version of all other sections Addresses comments on v0.2 that had reached an agreement by 12pm 2012-11-15
V0.9	2012-11-26	Rich Bruin; Neil Smith	K-INT	Incorporation of outstanding comments on v0.2 and new comments on v0.3
V1.0	2012-11-28	Rich Bruin; Neil Smith	K-INT	Incorporation of comments from project management board

Contents

1 Introduction

1.1 Purpose

The aim of this document is to set out the overall architecture for the Europeana Connection Kit (ECK) being developed as part of the Europeana Inside project. The different components within this architecture will then be defined and specified in such a way that they can be implemented by the various project partners and so that the implementations can be tested against the requirements in order to certify the developed tools as ECK compliant.

1.2 Scope

This document sets out the overall architecture for the ECK but is restricted to detailed specification of those components that have been determined to be part of Iteration 1 of development. Further versions of this deliverable will detail the components to be implemented for later iterations. These versions will be released as internal deliverables S2.6, S2.7 and S2.8 as explained below.

Iteration 1 of the ECK will consider all of the requirements from *D2.4: Functional Requirements* that have been designated as 'Must' haves with the exception of the actual data push and harvest interfaces onto Europeana and other aggregators (WFR.05.01 and WFR.05.02). This iteration is concerned with selecting and preparing the data for onward publication and is designed to be implemented with consideration of these specific requirements but it is expected that the actual implementation of requirements WFR.05.01 and WFR.05.02 will happen as part of the next iteration.

1.3 System Overview

1.3.1 System Requirements

The system requirements are set out in the *D2.4: Functional Requirements*. These requirements will be reviewed and developed for each iterative development.

1.3.2 System Deliverables

1.3.2.1 Iteration 1 (WP3)

The project deliverables which relate directly to Iteration 1 are D3.1 (Prototype), due in Month 13 (April 2013), and D3.2 (Codebase on GitHub) due in month 14 (May 2013).

1.3.2.2 Iteration 2 (WP3)

Iteration 2 will result in D3.3 (Prototype) and D3.4 (Technical Integration Progress Report) in Month 18 (September 2013). There is also an internal deliverable, S3.5 (Codebase on GitHub) in Month 19 (October 2013).

1.3.2.3 Iteration 3 (WP5)

Iteration 3 will produce internal deliverables S5.0 (Prototype) in Month 24 (March 2014) and S5.0.1 (Codebase on GitHub) in Month 25 (April 2014).

1.3.2.4 Iteration 4 (WP5)

Iteration 4, the final version will produce a number of deliverables. D5.1 (Production version) is due in Month 28 (July 2014) with D5.2 (Integration Status Report) and D5.3 (Technical Documentation) being available in Month 29 (August 2014) followed by D5.4 (Forward Plan) and internal deliverable S5.5 (Final Codebase on GitHub) in Month 30 (September 2014).

2 Design Considerations

2.1 Assumptions and Dependencies

- There are strong dependencies on Work Package 2 to provide updated requirements and specifications with this version of the technical specifications being based on v1.0 of D2.4 Functional Requirements and any further updates will be merged into later iterations of this document. Timely delivery of each iteration requires this specification document to be updated for each iterative release via internal deliverables S2.6 (due Month 14 – May 2013), S2.7 (due Month 20 – November 2013) and S2.8 (due Month 25 – April 2014). The final version of the technical specification will be published as D4.6, also in Month 25 (April 2014).
- As stated in the Description of Work, the seamless integration of the licensing framework for Europeana into the ECK is a key outcome of the project. A report will be produced by a technical legal expert who will produce an analysis and a set of recommendations for the representation of Europeana licenses in the ECK. The report will be appended to this deliverable when it becomes available. However, we do not expect this to materially change the deliverable.
- Work Package 4 is dependent on the outputs of Work Packages 3 and 5 to produce its deliverables. Therefore, the following functionality needs to be available for testing in the specified iteration:

o Data Selection and Transformation	Iteration 1
o Management Overview of status and data publication	Iteration 2
o Content re-ingestion from Europeana	Iteration 3

2.2 General Constraints

- Delivery dates for each iterative release of the software (see section 1.3.2 above) are fixed in the description of work and can only be changed with the approval of the project monitoring officer.
- Each functional area needs to have at least one reference open source implementation with the source code published on a widely accessible platform (e.g. GitHub).
- Many of the functional requirements are dependent on availability of functionality and services from Europeana. Unless Europeana is willing and able to make these available in the required timescales, it will not be possible to deliver the required functionality.
- The specification for each functional component should be technology neutral, i.e. it should be possible for the requirement to be met using a number of different technical approaches (e.g. java, .NET, C#, PHP, etc.).

2.3 Development Methodologies

The ECK development will follow an iterative approach. This allows some of the “easy wins” to be tackled first and ensures that a complete system is available for use as early in the project as possible. Later development iterations can then tackle more complex requirements and those requirements that could not be addressed until earlier requirements had been addressed. Allocation of development activity to each iteration is, however, constrained by the dependencies outlined in 2.1(bullet 2) above.

This approach means that new functionality can be given to users sooner than the more traditional waterfall approach, allowing them to find flaws while there is still time to correct them in later iterations. It also means that later iterations of the ECK design can be targeted at requirements which may not yet have become clear as part of the requirements elicitation process so far.

This document details those components of the ECK and their integration within the wider Europeana Inside environment that will be considered as part of Iteration 1 of the development process. Further development iterations will be considered by later versions of this document.

3 System Architecture

3.1 Overall Architecture

Due to the nature of the ECK and its integration within the many different CMS and aggregator systems throughout the Europeana Inside project there can be no one overall ‘system architecture’ in the traditional sense. Rather the ECK will be made up of a set of modular components that may or may not be implemented as standalone services in the Europeana Inside ecosystem rather than as a single monolithic whole. Some of these modules will actually come from existing functionality within CMS systems, others will be developed as part of this project and can be incorporated directly into or interfaced with the CMS or aggregation systems themselves and others might be existing third party components which can be used ‘as is’ or wrapped in a service later with appropriate API calls.

This lack of an overall architecture means that the ECK must be specified by the set of high level, functional and non-functional requirements that have been determined by previous stages of the project. In addition the interfaces and interactions between the ECK components and with external tools such as existing CMS components, vocabulary management systems, Europeana itself, etc. will make up a key part of the overall architecture.

Figure 1 is a representation of the overall architecture and environment in which the ECK will operate. Some of the functional requirements listed in D2.4 are to be provided by the CMS itself, while other parts are provided by external, shared modules. The connections between the components are of course as important as the individual modules themselves as they represent the interfaces presented by the different modules and the communications that are sent via these interfaces.

The figure depicts the overall architecture as consisting of an ECK ‘Core’ with additional modules as external to the ECK core. However, it is expected that many of the modules may actually be implemented within the core itself. They are simply presented this way in order to make the figure more easily understood. Equally it is expected that the aggregator and possibly even Europeana itself will also incorporate the ECK but again this duplication has been left out of the figure for simplification purposes.

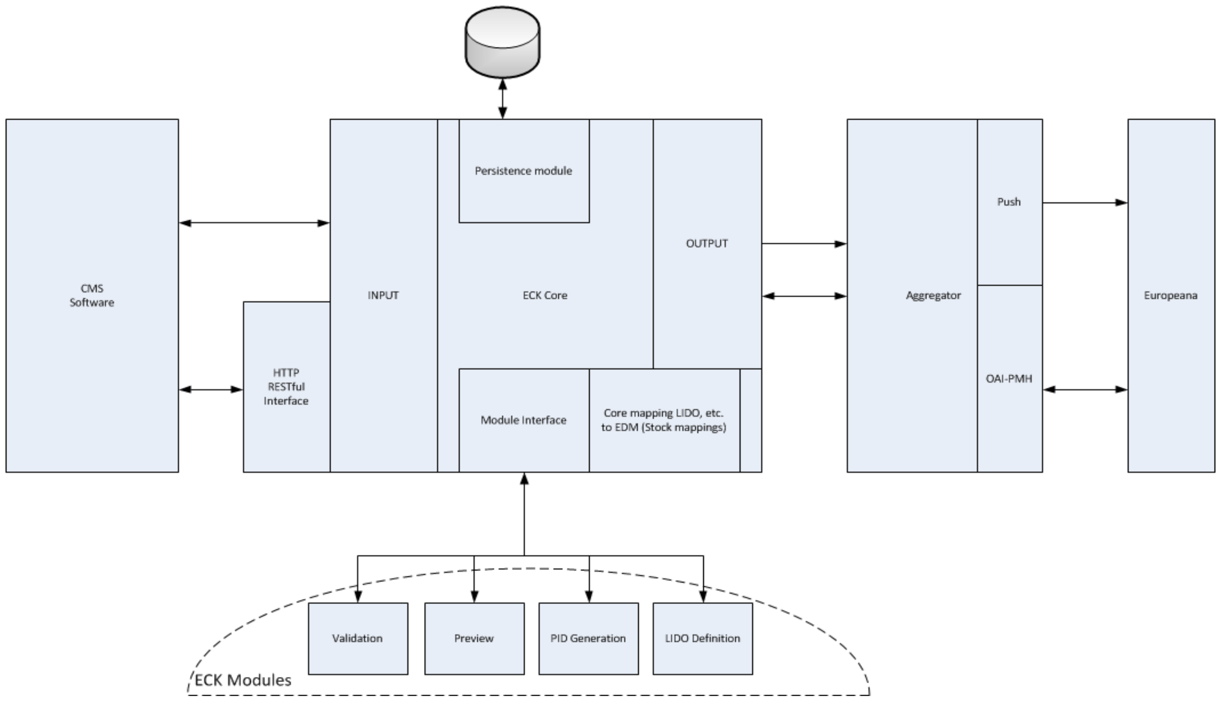


Figure 1: Representation of the overall ECK architecture and its communication with the CMS and aggregator / Europeana

3.2 General Implementation and Integration

The mechanism for integrating the ECK into (and deploying ‘alongside’) the different CMS solutions and aggregators will be dependent upon both the existing architectures and the software languages of these solutions. To that end we are proposing an event driven architecture at the library level for which we expect there to be multiple programming language implementations for the vendors who wish to integrate in this way.

If integration (or service exposure) is required at a higher level, we are also proposing wrapping these event driven libraries with HTTP RESTful calls so that features can also be accessed remotely. In turn these HTTP wrappers can be wrapped in language dependent SDKs which can themselves implement event based interfaces. These SDKs will then allow the CMS to use remote modules as event driven systems if desired. This communication architecture can be seen in Figure 2.

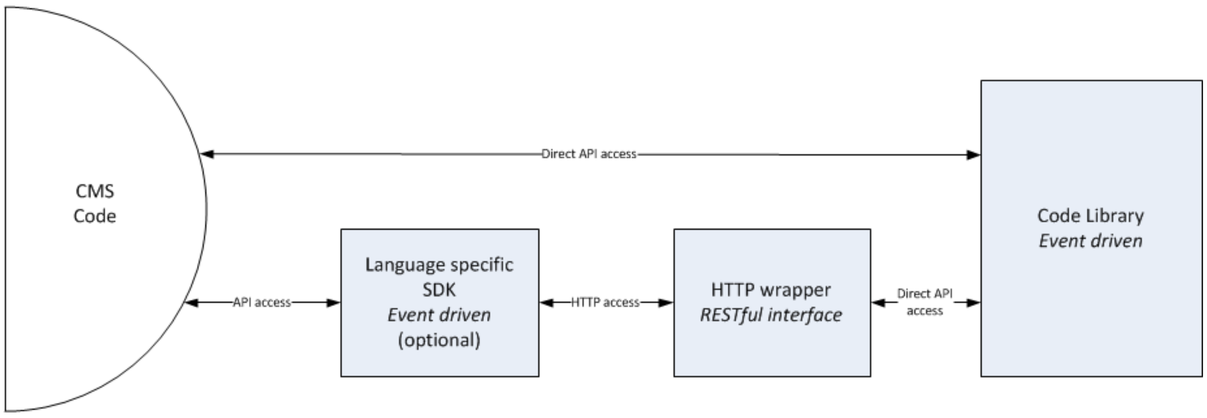


Figure 2: Overall implementation / deployment architecture enabling direct API access and/or (remote) HTTP access from the CMS code to the ECK code as required by each individual implementation scenario

3.3 Other Considerations

3.3.1 Introduction

This section details other considerations that have been taken into account when designing the technical architecture and requirements. These considerations are not examined in detail elsewhere in this document as they fall outside of the scope of Iteration 1 of the ECK development. However, it is important that they are considered in order to ensure that the overall architecture does not prevent their inclusion later.

3.3.2 High Level and Non-functional Requirements

Although they are not detailed explicitly in the following sections the ECK technical specifications have been developed to conform to both the high level and non-functional requirements as specified in the previous Europeana Inside deliverable document (D2.4).

3.3.3 Extensibility

Later iterations of the ECK must allow more complicated use cases (for example to implement WFR.03.19 – Semantic data enrichment) and must perform the actual data publication via Push to and/or Pull from Europeana and other aggregators. While these further requirements will not be implemented at this stage it is important that they are considered in the overall architecture, design and implementation so that other parts of the system are not implemented in such a way that they prevent their implementation.

3.3.4 Other Tools

There are already many existing tools that implement parts of the functionality required by the ECK. It has always been planned that where possible the ECK will make use of such existing tools in order to decide whether they can be used all, or in part to satisfy the requirements for the project. It is therefore proposed that one of the early tasks performed as part of the start of the ECK prototype development is a comprehensive review of the existing tools and to the extent they can be used or further developed as part of the ECK. These existing tools include:

- Mint - http://mint.image.ece.ntua.gr/redmine/projects/mint/wiki;
- Repox - http://repox.ist.utl.pt/;
- Europeana Content Checker - http://bit.ly/europeanaContentChecker (Documentation available from http://bit.ly/ContentCheckerDocs);
- Social History Services' PID tools - https://pid.socialhistoryservices.org/;
- Delving platform – http://www.delving.eu;
- Further tools listed at http://bit.ly/ToolUsage.

3.3.5 Other Required Modules

Additional functions must be implemented by the ECK in later iterations in order to ensure that the ECK supports the other requirements detailed in the previous deliverable document (*D2.4: Functional Requirements*). It is planned that most of these functions will be able to be implemented within the ECK in the form of additional modules rather than requiring direct change to the ECK core functionality itself.

One module that will definitely be required for Iteration 2 is the "Presentation Module". This module will be in charge of taking the created EDM documents and sending them on via push (possibly using the SWORD protocol[1]) or exposing an OAI-PMH server in order to allow record harvesting by Europeana and other upstream aggregators.

We will not be specifying this module fully at this stage since it falls outside the scope of Iteration 1. However, it is an important module as it obviously forms a key part of transferring data onto Europeana and it has wide ranging effects within the other parts of the ECK. For example, the persistence module, described later, will need to store information relating to publication and harvest events as well as to allow messages to be fed back from upstream data consumers to the original owner of the data relating to success or failure of data imports, etc. The datamodel described later takes this into account, but it is anticipated that there will need to be additions to this datamodel for future iterations.

[1] http://www.swordapp.org

4 Detailed Component Design

4.1 Introduction

The following sections set out the detailed design and specification of the different ECK system components. The order of these sections does not imply any order of invocation since this is likely to vary on a use by use basis.

4.2 CMS: ECK supporting functionality

4.2.1 Summary

This section represents functionality that must be provided within the main CMS in order for the CMS to be classed as ECK compliant for Iteration 1. It includes functions such as selecting and exporting records in the relevant metadata profile to the core ECK system as well as reporting on the state of record selection and export.

4.2.2 Assumptions

The process of configuring and setting up the CMS-ECK interface and the ECK functionality itself doesn't count towards implementing export as a 1-click process, but once configuration is complete a 1-click process should be possible.

Exactly how the different selection / reporting functionality is implemented is completely up to the CMS vendor involved, this specification just states what overall functions are required in order to be ECK compliant.

It is expected that the CMS may want to retrieve definitions of fields and explanations of their use from the ECK profile definition module. The messages from that module can then be overridden if they do not exactly fit the particular use case if required.

4.2.3 Functional Requirements Addressed

This component addresses:

- WFR.01.01 – Export management
- WFR.01.02 – Revision history
- WFR.01.03 – Notification changes to the ECK
- WFR.01.04 – PID management
- WFR.02.01 – Selecting multiple records
- WFR.02.02 – Selecting a single record
- WFR.02.03 – Selecting records based on values
- WFR.02.04 – Boolean operators
- WFR.02.05 – Indication of selected fields
- WFR.02.07 – Reuse saved queries (Increased to 'Must' from 'Could' due to feedback in technical partners meeting October 2012).
- WFR.03.01 – Automatic EDM mapping
- WFR.03.02 – Preview mapping
- WFR.03.03 – Editable mapping
- WFR.03.04 – Mapping feedback
- WFR.03.05 – Saving mapping
- WFR.03.06 – Field explanations
- WFR.03.07 – Automatic value insertion
- WFR.03.08 – Automatic thumbnail generation

- WFR.03.09 – Thumbnail selection
- WFR.03.12 – Metadata field on IPR digital object
- WFR.03.13 – Metadata field on IPR metadata
- WFR.03.14 – Metadata field on IPR preview
- WFR.03.15 – Mark mandatory fields
- WFR.03.16 – Choosing a default mapping
- WFR.03.20 – Conditional mapping
- WFR.03.21 – Nested or grouped mapping
- WFR.03.24 – Apply PID
- WFR.03.26 – Two level mapping
- WFR.04.03 – Edit validated fields
- WFR.05.01 – Automatic supply (Consideration of, but not implemented in this iteration)
- WFR.05.02 – Re-supply functionality for failed records (Consideration of, but not implemented in this iteration)
- WFR.06.03 – Update published records

4.2.4 Technical Design

4.2.4.1 Interfaces

The code for this section of the system is highly CMS specific with the exception of the interface with the ECK core. Therefore the only interface that is specified here is that the CMS is able to communicate with the interfaces exposed by the ECK core component.

4.2.4.2 Conformance Criteria

A CMS will be deemed to conform to this specification when it implements interfaces to satisfy each of the functional requirements listed above.

4.2.4.3 Implementation Expectations

Since this functionality is entirely contained within each CMS it is expected that each CMS will implement this section separately.

4.2.5 Outstanding requirements

The following requirements must be satisfied before this module can be fully implemented:

- Export format for metadata from CMS into ECK must be specified (LIDO profile[2], etc.)

[2] The development of this profile should consider http://bit.ly/HOPEMetadataStructure - The Common HOPE Metadata Structure, including the Harmonisation Specifications

4.3 ECK Core

4.3.1 Summary

This section represents the core of the ECK functionality as it is integrated into the various CMS systems. It is envisaged that this system will be directly included within the CMS itself and will accept data from the core CMS before passing it through the various ECK modules as appropriate and then onto consuming aggregators, etc. This section will also perform the stock mappings from the implemented metadata profiles into EDM ensuring consistency throughout the implementations and insulating the core CMS systems from changes to EDM.

4.3.2 Assumptions

Communications with processes that are likely to be long running should be event driven or poll based, while processes that will definitely return without delay should allow for direct result access.

This part of the system is exposed via an API that can be used directly by the main part of the CMS. An HTTP wrapper around the API can be exposed as an option if desired and should expose a RESTful interface.

The ECK core is stateful in that it must store information such as CMS ID, PID, upload information, harvest information (IP, date), etc. in order to allow reporting of this information by the CMS. It must also store the generated EDM representation in order to be able to support OAI-PMH harvesting if that option is implemented. However, it does not currently have to store any intermediate representations of the metadata as this is the domain of the CMS itself although this may change in response to requirements in later ECK iterations.

The state stored by the ECK can be in a separate data store or within the main CMS data store as desired by the CMS meaning that if the CMS vendor wishes they can access and modify harvest, etc. data directly without having to use the relevant functionality within the ECK core in order to better support their interfaces.

Due to the use of fixed profiles for data input into the ECK Core it is possible to discover and retrieve the generated PID for a document from the document itself. This means that this is not required as an additional argument when importing new or updating existing records within the ECK.

The choice of technology for mappings from the format(s) received by the ECK core as input and EDM is implementation specific and need not be defined here as long as it supports all of the required mapping functionality.

4.3.3 Functional Requirements Addressed

This section addresses:

- WFR.03.01 – Automatic EDM mapping – directly;
- All of the functional requirements that are addressed by the separated ECK modules detailed here;
- Consideration of requirements WFR.05.01 (Automatic supply) and/or WFR.05.02 (Re-supply functionality for failed records) to export the generated EDM onto aggregators, etc. although their implementation will not occur as part of this iteration.

4.3.4 Technical Design

4.3.4.1 Interfaces

The method exposed by the core and the functionality exposed here is expected to change as part of later iterations. However, many of these additional functions will be implemented as calls to modules that the ECK core wraps and proxies calls to and so should not require too many changes to the ECK Core structure itself.

The ECK Core will expose the following methods as interfaces to the CMS as part of Iteration 1:

- List available functions (including those from modules);
- Call methods on modules;
- Import record into or update existing record in ECK (This is likely to be expanded to allow publication, etc. in later iterations.

Figure 3 shows how each of these interfaces should behave.

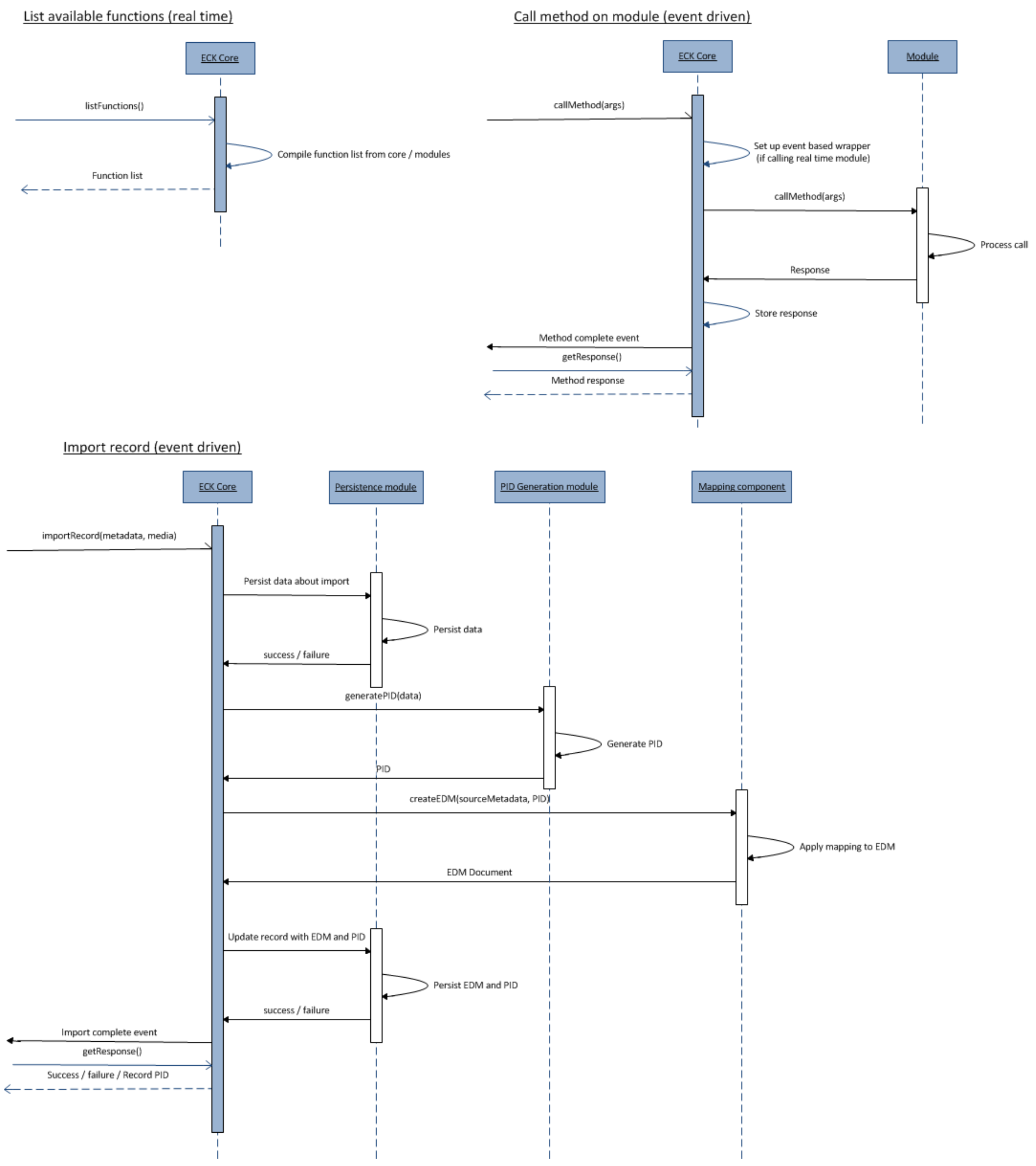


Figure 3: Interfaces to be implemented and exposed by the ECK Core component

4.3.4.2 Conformance Criteria

In order to conform to these specifications the module must implement each of the mandatory interfaces detailed here. In addition all modules that are to be integrated with the ECK core should be fully exposed by the ECK core for use by the CMS which submitted the request.

4.3.4.3 Implementation Expectations

It is expected that there will be one implementation of the ECK core which will then be wrapped in a RESTful HTTP interface to allow integration with tools written in other languages. This HTTP interface will not implement an event based interface but will rely on the caller polling for updates or on language specific SDK implementations which in turn wrap the HTTP interface if these are required by the various partners.

Future iterations of this component may require that multiple mappings are performed by the ECK in order to transform from the input data format to the output EDM (or possibly other output data formats). In addition it is possible that the mapping to be applied may not be chosen automatically by the software and that it must be specified by the caller. Although these are not requirements for this iteration they should be taken into account when implementing the component to ensure that they are possible for later iterations.

4.3.5 Outstanding requirements

The following requirements must be satisfied before this module can be fully implemented:

- Import format for metadata must be specified (LIDO profile for Iteration 1. Developing EAD and MARC profiles at a later stage)
- Report formats from the various modules must be specified

4.4 Metadata profile definition module

4.4.1 Summary

This module will allow for the provision of multilingual definitions for the fields in the different profiles that are implemented. The caller will be able to request all definitions for a profile in a given language or the definition of a single field in a given language.

In addition the module will enable the lookup of meaningful multilingual error definitions and guidance on steps to take to avoid them for use when explaining validation and other errors that have occurred elsewhere in the ECK system. For example validation errors can be returned as codes from the validation module and they can then be dereferenced in the appropriate language for the user using this module.

4.4.2 Assumptions

None

4.4.3 Functional Requirements Addressed

This module addresses:

- WFR.03.06 – Field explanations

4.4.4 Technical Design

4.4.4.1 Interfaces

The module will expose the following methods as interfaces for use by other parts of the ECK:

- List languages (all languages that are available at all in the system);
- List profiles (all available profiles and the languages in which they are available);
- Request definitions (all definitions for a profile for a language);
- Request individual definition / guidance;
- Request error definitions (all error definitions for a language);
- Request individual error definition / guidance.

Figure 4 shows how each of these interfaces should behave.

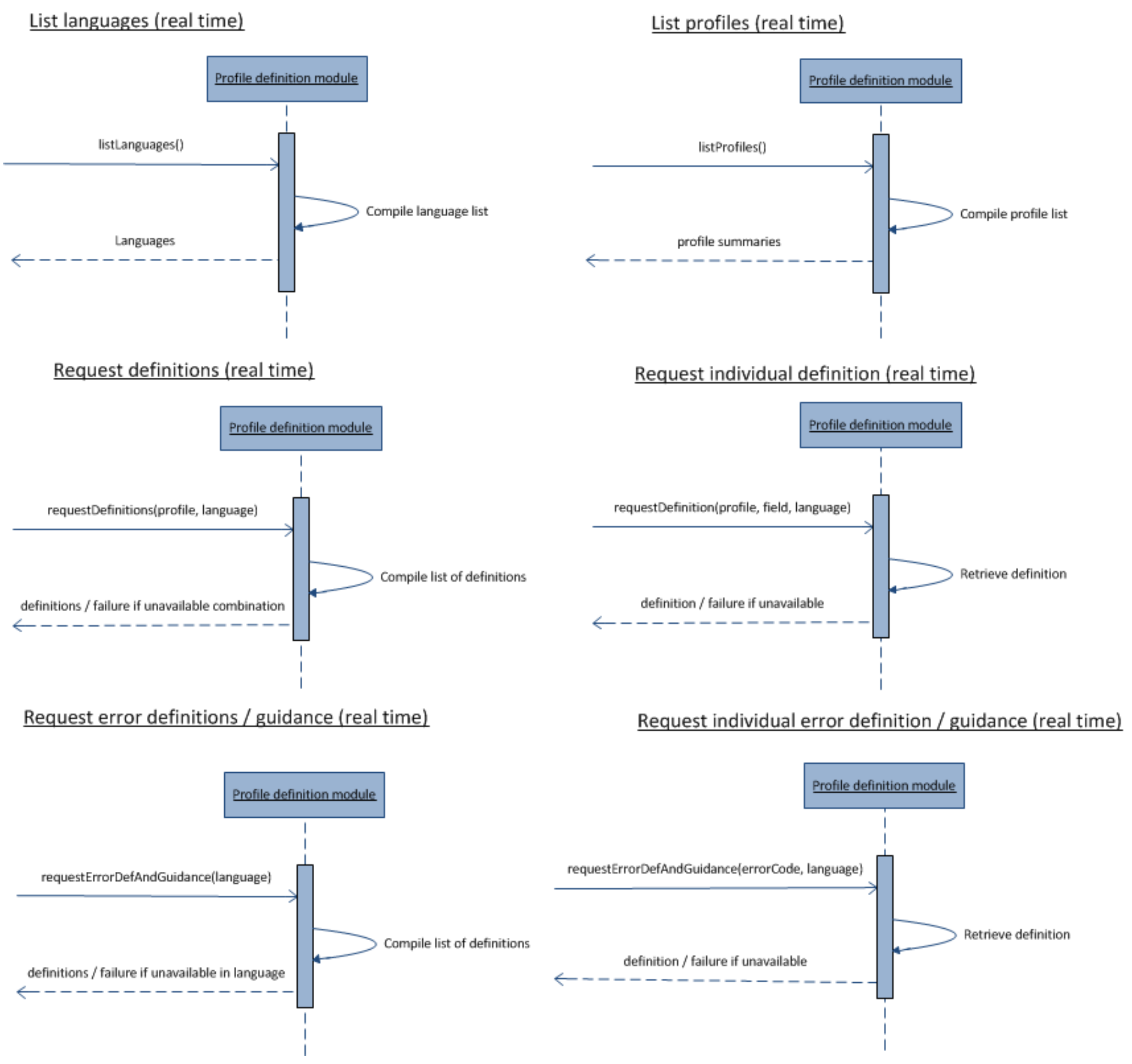

List languages (real time)
Profile definition module
listLanguages()
Compile language list
Languages
List profiles (real time)
Profile definition module
listProfiles()
Compile profile list
profile summaries
Request definitions (real time)
Profile definition module
requestDefinitions(profile, language)
Compile list of definitions
definitions / failure if unavailable combination
Request individual definition (real time)
Profile definition module
requestDefinition(profile, field, language)
Retrieve definition
definition / failure if unavailable
Request error definitions / guidance (real time)
Profile definition module
requestErrorDefAndGuidance(language)
Compile list of definitions
definitions / failure if unavailable in language
Request individual error definition / guidance (real time)
Profile definition module
requestErrorDefAndGuidance(errorCode, language)
Retrieve definition
definition / failure if unavailable

Figure 4: Interfaces to be implemented and exposed by the profile definition module

4.4.4.2 Conformance Criteria

This module will be deemed to conform to the specification when it exposes each of the above defined interfaces for use by other modules and the ECK core / CMS components.

4.4.4.3 Implementation Expectations

It is expected that there will be a single implementation of this module that will be able to be shared by all users of the ECK but there may be more than one installation in order to allow for different deployment scenarios.

4.4.5 Outstanding requirements

The following requirements must be satisfied before this module can be fully implemented:

- Each metadata format profile must be specified;
- The list of spoken languages which are to be supported must be defined;
- For each field in the profiles definitions and examples of use must be generated in each required language;
- For each specified error code (particularly possible validation errors) a meaningful error message and guidance on what it means and how to rectify it in the user’s data and mappings must be generated for each required language.

It will be necessary to translate the various messages, etc. to be generated as a part of the project. These translations should be shared as much as possible throughout the project. Therefore efficient generation of the translations is very important to ensure that multiple partners do not perform the same translation unless there is a good reason for this to happen. Existing translation tools such as Transifex[3] should be investigated to see if they can be used.

4.5 Persistence module

4.5.1 Summary

This module enables the ECK core to communicate with the required data store. This is required for two reasons:

- To enable the ECK to save its state in non-standard database system if required; And
- To allow the ECK to use the CMS datastore without worrying about conflicts with existing data structures and to allow relevant foreign keys to be linked as appropriate.

4.5.2 Assumptions

None

4.5.3 Functional Requirements Addressed

This module addresses:

[3] http://www.transifex.com

- WFR.01.01 – Export management
- WFR.01.02 – Revision history (possibly)
- WFR.01.04 – PID management (as part of use of the PID generation module)
- WFR.05.02 – Re-supply functionality for failed records
- Additionally it will directly support each of the event / poll based communications which require state to be stored before passing it onto the CMS, etc.

4.5.4 Technical Design

4.5.4.1 Abstract datamodel

Figure 5 represents the abstract datamodel as required by the current technical specifications. It is expected that there will be some extension required (for example to support revision histories) and possibly implementation specific fields will be added. It is also expected that individual CMS providers may want to link these entities to existing entities in their existing datastore although that is not a requirement of these specifications.

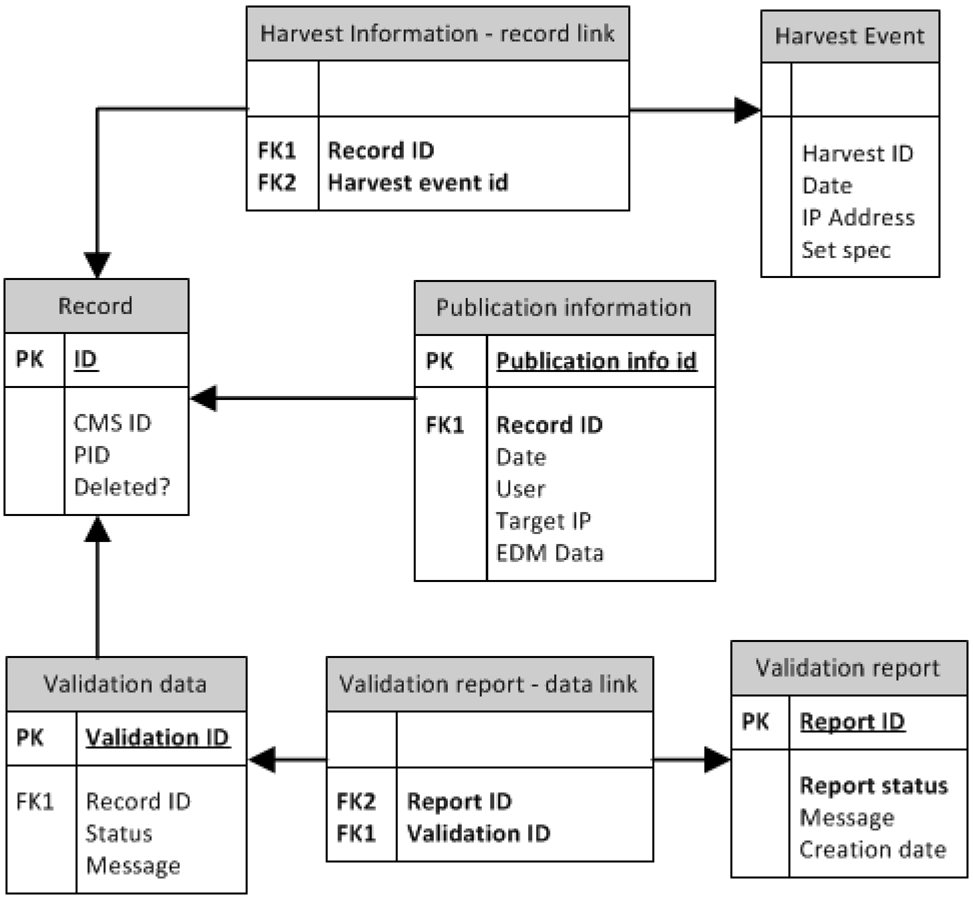


Figure 5: Abstract persistence module datamodel

4.5.4.2 Interfaces

The following methods are required to be exposed via the module's interfaces for use within the ECK:

- Record entity:
  - Lookup by ID, CMS ID or PID
  - Create new
  - Save / update / delete
- Publication information entity
  - Lookup by ID, Linked record ID
  - Create new
  - Save / update / delete
  - Link to record
- Validation report
  - Lookup by ID
  - Create new
  - Save / update / delete (with relevant cascades)
- Validation data
  - Lookup by ID, Report ID, Record ID and by Record ID and Report ID combined
  - Create new
  - Save / update / delete
  - Link to validation report
  - Link to record
- Harvest event
  - Lookup by ID, IP Address, Date, Set spec
  - Create new
  - Save / update / delete
  - Link to record

4.5.4.3 Conformance Criteria

In order to conform to these requirements the module should:

- Fully model the data specified in the entity-relationship diagram above (although the precise model need not match the diagram exactly);
- Implement and expose each of the specified interfaces for use by other parts of the ECK.

4.5.4.4 Implementation Expectations

It is expected that there will be multiple implementations of the persistence module with at least one implementation per implementation language. It is also expected that CMS vendors will either want to extend the standard module to interface with their existing datamodel or that they will write their own implementation to communicate with non-standard database systems.

4.5.5 Outstanding requirements

The following requirements must be satisfied before this module can be fully implemented:

- None

4.6 PID Generation module

4.6.1 Summary

This module is in charge of creating PIDs for those records which do not already have identifiers which are globally unique and persistent. It will allow creation of PIDs based on institution URL (or some other identifier if no URL is available), record type and record accession number (or other suitable local identifier), combining these into a PID. Reverse lookup will also be supported allowing a generated PID to be separated and the constituent parts returned.

It is intended that the PID generated here can be used to drive a co-referencing service that allows links to be made between CMS IDs and other external IDs such as those exposed as part of an OAI-PMH server, generated by Europeana or other aggregators, etc.

4.6.2 Assumptions

This whole module is only required if the CMS does not already contain records with suitable PIDs or a field that can be used directly as the PID

The module can work in two different use cases:

- “Built in” where it is a part of the ECK for a particular install and as such can be configured with the installed institution URL and so requires less information when generating a PID;
- “Shared / remote” where it is accessed remotely or is shared between different institution’s installations. In this case the institution URL must be specified in order to generate the PID.

4.6.3 Functional Requirements Addressed

This module address:

- WFR.01.04 – PID management
- WFR.03.24 – Apply PID

4.6.4 Technical Design

4.6.4.1 Interfaces

The module will expose the following methods as interfaces for use by other parts of the ECK:

- Configure (for “built in” use case)
- Show configuration (for “built in” use case)
- Generate PID
- Reverse lookup PID (return constituent parts)

Figure 6 shows how each of these interfaces should behave.

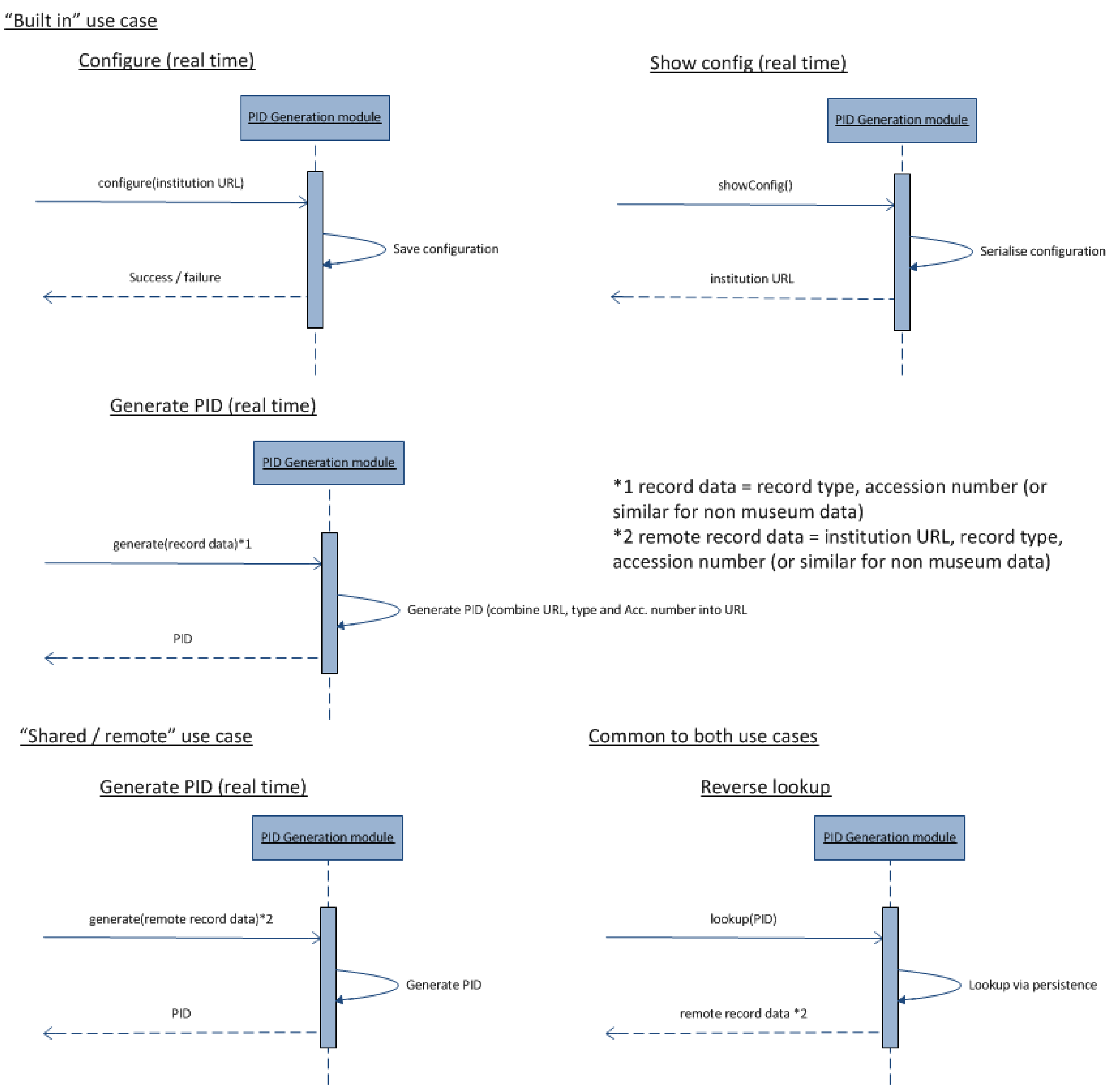


Figure 6: Interfaces to be implemented and exposed by the PID Generation module

4.6.4.2 Conformance Criteria

Implementations of this module will conform to the specification when they fully expose each of the above specified interfaces for use.

4.6.4.3 Implementation Expectations

It is expected that there will be multiple installations of this module either on an installation by installation basis (the "built in" use case), or per CMS, or shared between the projects (the "remote / shared" use case).

The module should be written in such a way that it is possible to 'plug in' external PID generators at a later date. This will allow the use of existing services such as handle[4] or doi[5]. The module should also allow for future cases where the institution involved does not have its own URL, or uses non-unique accession numbers, etc.

It is also expected that the implementation(s) of this module will take into account existing recommendations and standards for ID generation in this field. Examples of such standards and recommendations are:

- Europeana Persistent Identifiers task force - http://bit.ly/PIDTaskforce;
- ISIL - http://biblstandard.dk/isil/;
- MuseumID - http://museumid.net;
- CIDOC ID recommendations - http://bit.ly/CIDOC-IdRecommendations.

4.6.5 Outstanding requirements

The following requirements must be satisfied before this module can be fully implemented:

- None

4.7 Preview module

4.7.1 Summary

This module allows for the generation of preview web pages to show how the user's data will look when imported into Europeana (and optionally other targets e.g. intermediate aggregators, etc.). Given a single or set of metadata records and related media files the module will populate template pages for a sample hit list and record details page including thumbnails, etc. These previews will be packaged into a ZIP archive and returned to the caller or in the case where a web presence for the module is required the preview can be hosted directly and a link to the resource returned to the caller.

4.7.2 Assumptions

The module does not require a web presence to allow the previews to be viewed in the case where the calling system (CMS) requires a bundle to be returned. However the default behaviour is that the preview is hosted for access via a web browser. It is anticipated that some project partners will prefer the ability to expose the preview directly on a website to be linked to, while others will prefer full control and as such require the preview bundle to be returned. Therefore it is believed that ECK deployments will differ in their usage scenario. The module should also be able to act as a façade to existing preview services such as the Europeana content checker if they provide the functionality required.

4.7.3 Functional Requirements Addressed

This module addresses:

- WFR.06.01 – Preview presentation Europeana

[4] http://www.handle.net/

[5] http://www.doi.org/

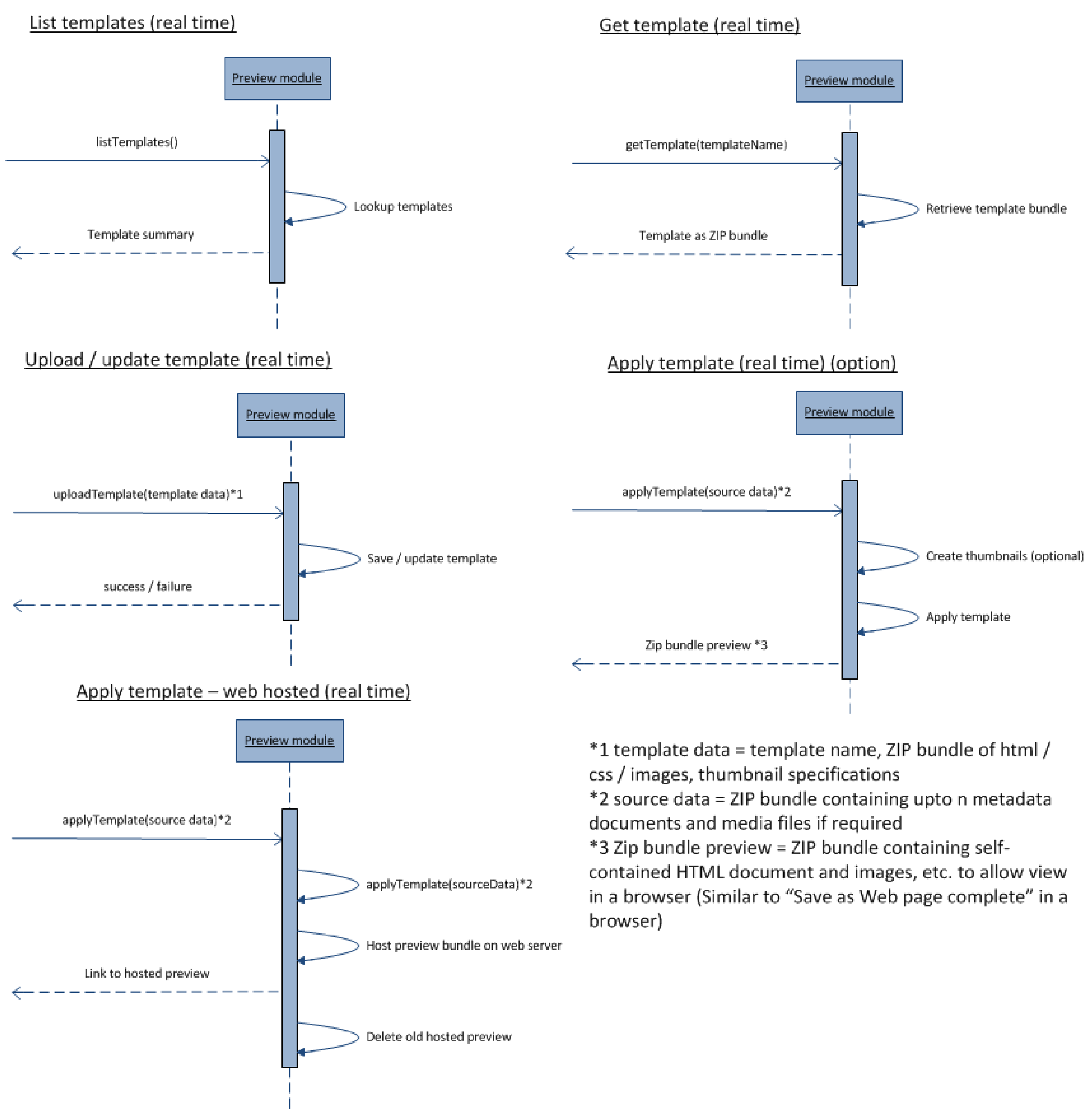
4.7.4 Technical Design

4.7.4.1 Interfaces

The module will expose the following methods as interfaces for use by other parts of the ECK:

- List preview templates
- Get preview template
- Upload / update preview template
- Apply preview template and return bundle (optional)
- Apply preview template with web presence

Figure 7 shows how each of these interfaces should behave.


List templates (real time)
Preview module
listTemplates()
Lookup templates
Template summary
Get template (real time)
Preview module
getTemplate(templateName)
Retrieve template bundle
Template as ZIP bundle
Upload / update template (real time)
Preview module
uploadTemplate(template data)*1
Save / update template
success / failure
Apply template (real time) (option)
Preview module
applyTemplate(source data)*2
Create thumbnails (optional)
Apply template
Zip bundle preview *3
Apply template – web hosted (real time)
Preview module
applyTemplate(source data)*2
applyTemplate(sourceData)*2
Host preview bundle on web server
Link to hosted preview
Delete old hosted preview

*1 template data = template name, ZIP bundle of html / css / images, thumbnail specifications
*2 source data = ZIP bundle containing upto n metadata documents and media files if required
*3 Zip bundle preview = ZIP bundle containing self-contained HTML document and images, etc. to allow view in a browser (Similar to "Save as Web page complete" in a browser)

Figure 7: Interfaces to be implemented and exposed by the preview module

4.7.4.2 Conformance Criteria

In order to conform to these requirements any implemented preview module must expose each of the mandatory interfaces detailed above. The optional interface can also be implemented, but without the other interfaces the module does not conform to these specifications

4.7.4.3 Implementation Expectations

It is expected that there may only be one implementation of the preview module, or possibly one per implementation language in order to make the maintenance of the previews themselves simpler. It is also expected that the implementation may act as a façade around existing preview services such as the Europeana content checker if they provide the required functionality.

4.7.5 Outstanding requirements

The following requirements must be satisfied before this module can be fully implemented:

- Input format for metadata must be specified (LIDO profile, etc.);
- Preview template format which allows the creation of actual preview templates must be designed, specified and documented;
- The Europeana Content Checker should be fully evaluated before implementation in order to decide whether it can be made use of within the project rather than implementing the module from scratch.

4.8 Validation module

4.8.1 Summary

This module provides validation functionality for the ECK. It receives one or more metadata documents and media files if appropriate. The module then performs the following types of validation on the provided data:

- Schema validation against the specified metadata profiles;
- Checks that media exists if referenced;
- Checks that media is referenced if it exists;
- Check that referenced media is of a suitable size to fit with Europeana guidelines[6];
- Checks field contents for things that can't easily be checked by schema validation including URI fields contain URIs, etc.;
- Option – Checks against 'style guidelines' for the profile – "rules of thumb" checks that titles aren't too long, etc. which may suggest that data from the wrong field has been mapped.

A validation report is then returned to the calling system which can be parsed and presented to the user and the persistence layer is used to annotate the record with the validation results. This report will contain validation error codes which can then be dereferenced for each required language using the profile definition module specified in Section 4.4.

4.8.2 Assumptions

None

4.8.3 Functional Requirements Addressed

This module addresses:

- WFR.04.01 - Validation
- WFR.04.02 – Feedback on validation
- WFR.04.04 – Automatic license validation
- WFR.04.06 – Align validation

4.8.4 Technical Design

4.8.4.1 Interfaces

The module will expose the following methods as interfaces for use by other parts of the ECK:

- One by one validation
- Batch validation

Figure 8 shows how each of these interfaces should behave.

[6] http://bit.ly/europeanaImageGuidelines

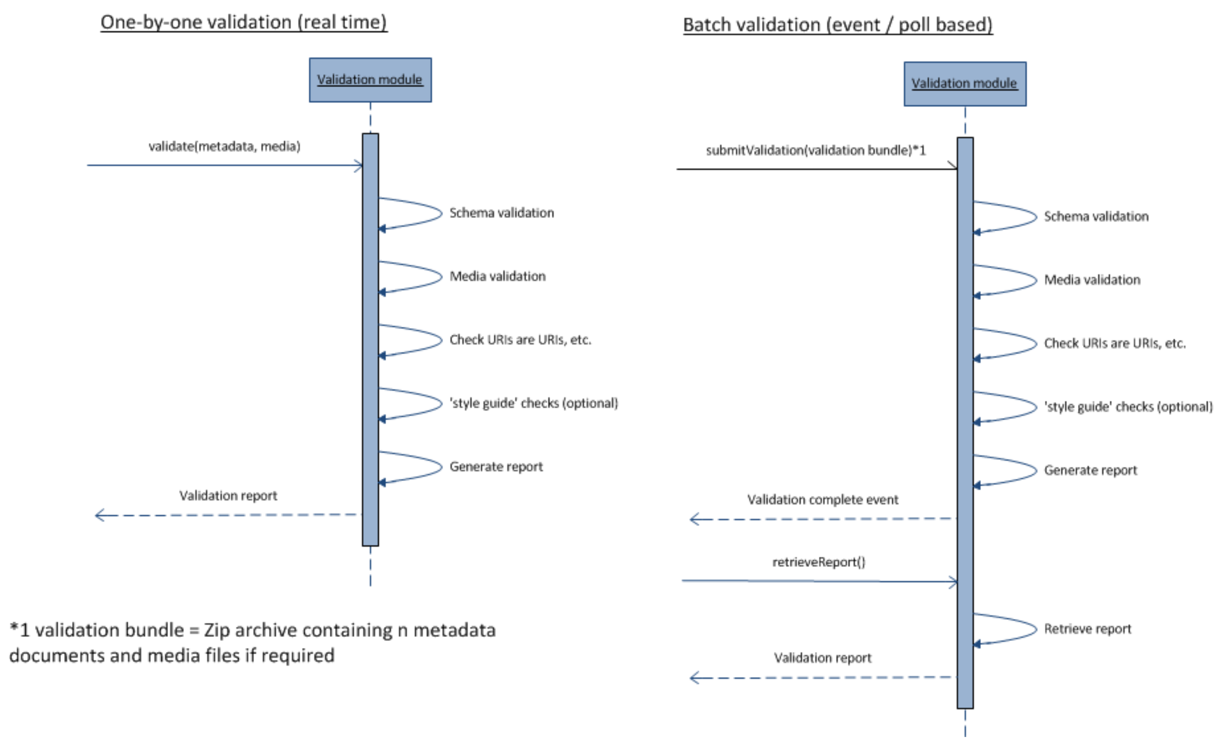

One-by-one validation (real time)
Validation module
validate(metadata, media)
Schema validation
Media validation
Check URIs are URIs, etc.
'style guide' checks (optional)
Generate report
Validation report
*1 validation bundle = Zip archive containing n metadata documents and media files if required
Batch validation (event / poll based)
Validation module
submitValidation(validation bundle)*1
Schema validation
Media validation
Check URIs are URIs, etc.
'style guide' checks (optional)
Generate report
Validation complete event
retrieveReport()
Retrieve report
Validation report

Figure 8: Interfaces to be implemented and exposed by the validation module

4.8.4.2 Conformance Criteria

Implementations of this module will be deemed to conform to the specification when they fully implement each of the interfaces detailed above.

4.8.4.3 Implementation Expectations

It is expected that there will be one implementation which is then wrapped and exposed using an SDK in each implementation language. This will keep the number of variations to a minimum and will help to ensure that validation is consistent across ECK implementations in general.

4.8.5 Outstanding requirements

The following requirements must be satisfied before this module can be fully implemented:

- Input format for metadata must be specified (LIDO profile, etc.);
- XML Schema for the profile must be implemented (for schema validation);
- Conditional and specialist validation must be specified (i.e. validation that cannot be represented using XML Schema, etc.);
- ‘Style guidelines’ must be determined (optional).

5 Current Planning & Issues

5.1 Development Schedule

The planned development schedule is set out in Annex 1. General constraints and dependencies are outlined in the relevant sections above.

5.2 Work Package and Project Management

Knowledge Integration (K-INT) is work package leader for work packages 3 and 5. During the timescale of each work package, all participating partners will be asked to supply a brief (no more than 1 side of A4) monthly progress report to K-INT. A template for this report will be supplied at the start of work package 3.

Overall management of the project is the responsibility of the Management Board, coordinated by Collections Trust. Sign off of all internal and external deliverables will be managed using the procedures set out in the project initiation document.

Where multiple partners are working collaboratively on a software component, a task coordinator will be appointed to oversee the development

5.3 Resourcing

It is expected that each partner will allocate the resources set out in the Description of Work to each work package. Any expected variances from this should be reported via the monthly report. Moving resource between work package 3 and work package 5 is permitted but plans to do so should be notified via the monthly report.

5.3.1 Technical Partners

It is the responsibility of each technical partner to decide on its own development priorities. In particular, for each technical component, technical partners must decide whether to implement the component. If it is to be implemented the partner must decide whether to:

- Provide a local implementation;
- Work with others on a common implementation;
- Adopt or adapt a common implementation developed by others;
- Adopt or adapt an existing third party component.

The decision on the approach taken should be communicated to the work package leader via the monthly report.

As a general rule of thumb, technical partners are advised that an overall distribution of resource might typically be as follows:

• Iteration 1	60-70% of WP3 resources
• Iteration 2	30-40% of WP3 resources
• Iteration 3	50-60% of WP5 resources
• Iteration 4	40-50% of WP5 resources

5.3.2 Content Partners

Content partners should liaise closely with their associated technical partner to plan their input to work packages 3 and 5. Any content partner not associated with a technical partner should liaise with the work package leader. Content partners will be responsible, amongst other things, for verifying that the software components delivered by technical partners meet the specified conformance criteria.

5.4 Issues

In producing this version of the technical specification, the following issues have arisen which will need to be addressed in subsequent iterations:

OI-1. The proposed LIDO profile is only appropriate for museum objects. Is it within the scope of this project to develop similar profiles of EAD for archival finding aids and MARC for bibliographic objects and what about other object types e.g. 3D and archaeological?[7]

OI-2. The architecture currently proposed is event driven. However this can cause problems in implementation when systems have to wait for other systems to respond. Would it be possible to make being event / poll driven optional?

OI-3. WFR.04.03 (Edit validated fields) calls for editing of output data after mapping has taken place. However this raises issues such as where the modified records are stored and how the system can remove the need for these edits to be made each time an export is made. Does this really make sense? Wouldn't it be better to be able only to make changes to the source data (i.e. in the CMS) and/or the mappings?

OI-4. Does it make sense to enforce that in order to be labelled as ECK compliant that technical partners should use at least one shared module that has been developed within the project or would it be acceptable for each partner to completely implement everything on their own based on the specifications?

OI-5. In the event that there is not an agreed consensus on the technical plan for a component at the end of the time allocated to the specification phase for an iteration it is proposed that the component be delayed to the next iteration. This should allow for a consensus to be achieved but can only happen for those components that are not required by other components within that iteration.

OI-6. To what extent should user management be implemented within the ECK? There are high level requirements in D2.4 relating to identification and authentication, and authorisation but there are no functional requirements currently relating to user management, etc. For this iteration we can safely leave out user management but to what extent should future iterations of the requirements and ECK consider users and how much of this can be left to the CMS or aggregation systems?

OI-7. Are Europeana still willing to trial data push with us? How much influence do we have on the technologies and standards that are used? Could SWORD be used?

OI-8. Do Europeana plan to change how they deal with records where there are multiple media files? The current system requires one metadata record per media file, but this does not always make sense.

[7] http://bit.ly/HOPEMetadataStructure - The Common HOPE Metadata Structure, including the Harmonisation Specifications – Work on specifying the LIDO (and other) profiles should consider this previous work where possible

OI-9. Do we expect an ECK compliant implementation to include all of the functions that are ‘Must’s in D2.4 Functional Requirements or should some functions be optional? For example, not all CMS vendors will want to expose mapping editing as they believe that this is not required for their users.

OI-10. Should there be a CMS independent implementation of all features? Or maybe an implementation tied to an open source CMS (e.g. CollectionSpace)?

OI-11. Are the conformance criteria sufficient for labelling a system as ECK compliant? Who should be responsible (e.g. should there be a formal ‘kitemarking’ procedure)? How do we ensure that this is sustainable after the end of the project and accessible by vendors who are not project partners?

OI-12. Appropriate licensing terms need to be agreed for all components of the ECK in order to allow their integration into (closed-source) CMSs. The DOW mentions GPLv3. Is that possible? What other licenses should be considered?

6 Acceptance & Sign Off

6.1 Acceptance

Software components must meet the specified conformance criteria to be accepted. Content partners are responsible for verifying that the conformance criteria have been met. Where appropriate, a test harness or detailed testing procedure may be developed to ensure consistency.

6.2 Sign Off

All internal and external deliverables will be signed off by the project board and the coordinating partner using the procedures specified in the project initiation document.

7 Glossary of terms

[term]	[definition]
CMS	Collection Management System – A system used for managing the data held about the objects held by museums and other institutions
CMS ID	The identifier for a resource as generated by the CMS
CP	Content Provider – A Europeana Inside project partner who is involved in the project in order to facilitate contribution of their data into Europeana
CRUD	Create Read Update Destroy – The main functions required in order to save and maintain data as persistent storage
ECK	Europeana Connection Kit – The system being specified here
EDM	Europeana Data Model – The data model that Europeana uses for data it ingests, etc. http://www.europeana.eu/schemas/edm/
HTTP	HyperText Transfer Protocol – the standard protocol used when communicating with web servers
LIDO	Lightweight Information Describing Objects – A metadata standard for representing data about objects http://www.lido-schema.org
OAI-PMH	Open Archives Initiative – Protocol for Metadata Harvesting. A standard that sets out how metadata records can be served and harvested as a means of sharing
PID	Persistent IDentifier – an identifier that will always represent the resource to which it refers and is unique
REST / RESTful	Representational State Transfer is a style of software architecture that allows for access to services exposed on remote servers typically via the web. A system is said to be RESTful if it conforms to REST principles. A full discussion of REST can be found at http://en.wikipedia.org/wiki/Representational_state_transfer
SDK	Software Development Kit – A set of software tools that allow for the implementation of functionality within other tools.
TP	Technical Partner – A Europeana Inside project partner who is involved in the project in order to contribute to and develop the ECK system (at least for the purposes of this document)
URL	Uniform Resource Locator – a string that serves as an identifier for a resource on the Internet and provides for the capacity to locate the resource
WFR.xx.xx	Functional requirement as defined in the previous Europeana Inside deliverable D2.4 Functional Requirements.
WPx	Work Package x – A specific stage of the Europeana Inside project.

Annex 1 - Europeana Inside Iterative Development Plan

Month	DOW	Revised Plan	
		WP 2 and WP 3 and WP5	WP 4
Iteration 1			
07	D2.4 Functional Requirement D2.5 Technical specification	D2.4 Functional Requirement	
08		D2.5 Tech Architecture & detailed specification	
09			
10			
11			
12			
13	D3.1 EUROPEANA INSIDE Prototype -Iteration 1	D3.1 Prototype	
14	D3.2 EUROPEANA INSIDE Codebase	D3.2 Codebase on GitHub	
15			
16	D4.1 Control Export Evaluation Report		D4.2 Content Export Schedule D4.1(v1) Control Export Evaluation Report

Month	DOW	Revised Plan	
		WP 2 and WP 3 and WP5	WP 4
Iteration 2			
14		S2.6 Detailed specification	
15			
16			
17			
18	D3.3 EUROPEANA INSIDE Management Interface D3.4 Technical Integration Report	D3.3 Prototype D3.4 Technical Integration Progress Report	
19		S3.5 Codebase on GitHub	
20			D4.1(v2) Control Export Evaluation Report
Iteration 3			
20	D4.2 Content Export Schedule	S2.7 Detailed specification	
21	D4.3 Export Evaluation Report D4.4 Content Re-ingestion Report D4.5 Summative Evaluation Report D4.6 Revised Technical Specification		
22			
23			
24		S5.0 Prototype	
25		S5.0.1 Codebase on GitHub	
26			D4.3(v1) Export Evaluation Report D4.4 Content Re-Ingestion Report D4.5(v1) Summative Evaluation Report

Month	DOW	Revised Plan	
		WP 2 and WP 3 and WP5	WP 4
Iteration 4			
25		S2.8 Detailed specification (same as D4.6 Revised Technical Specification?)	D4.6 Revised Technical Specification (same as S2.8 Detailed specification?)
26			
27			
28	D5.1 Production version	D5.1 Production version	
29	D5.2 Integration Status Report D5.3 Technical Documentation	D5.2 Integration Status Report D5.3 Technical Documentation	D4.3(v2) Export Evaluation Report D4.5(v2) Summative Evaluation Report
30	D5.4 Forward Plan	D5.4 Forward Plan S5.5 Codebase on GitHub	

Key:

Dx.x	Formal deliverable as specified in DOW
Sx.x	Software related task, not in DOW but required for iterative development