Grant Agreement 297292

# *EUROPEANA INSIDE*

# Technical Specification

| | |
|---|---|
| **Deliverable number** | D4.6 |
| **Dissemination level** | Public |
| **Delivery date** | April 2014 |
| **Status** | V5.0 |
| **Author(s)** | K-INT |

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

## Revision History

| Revision | Date | Author | Organisation | Description |
|---|---|---|---|---|
| V0.1 (D2.5) | 2012-10-31 | Neil Smith | K-INT | Document structure |
| V0.2 (D2.5) | 2012-11-09 | Rich Bruin<br>Rob Tice<br>Gill Osguthorpe<br>Ian Ibbotson | K-INT | Initial version of architecture and component requirements sections |
| V0.3 (D2.5) | 2012-11-16 | Rich Bruin<br>Neil Smith<br>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 (D2.5) | 2012-11-27 | Rich Bruin<br>Neil Smith | K-INT | Incorporation of outstanding comments on v0.2 and new comments on v0.3 |
| V1.0 (D2.5) | 2012-11-28 | Rich Bruin<br>Neil Smith | K-INT | Incorporation of comments from project management board |
| V1.5 (S2.6) | 2013-04-05 | Chas Woodfield<br>Rob Tice<br>Neil Smith | K-INT | Update to architecture to cover all four iterations of development |
| V2.9 (S2.8) | 2013-12-10 | Neil Smith<br>Chas Woodfield | K-INT | Updated in preparation for iteration 3. Issues relating to data push and content re-ingestion need resolving prior to v3.0 being issued |
| V3.0 (S2.8) | 2014-01-16 | Neil Smith<br>Chas Woodfield | K-INT | Updated version circulated to technical partners |
| V4.0 (D4.6) | 2014-04-21 | Neil Smith<br>Chas Woodfield | K-INT | Updated version circulated to technical partners |
| V5.0 (D4.6) | 2014-04-29 | Neil Smith<br>Chas Woodfield | K-INT | Final version incorporating partner comments |

**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This deliverable contains original unpublished work except where clearly indicated otherwise.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.

## Contents

# 1. Introduction

## 1.1. Purpose

The aim of this document is to set out the overall architecture for the Europeana Connection Kit (ECK) which was 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 and includes detailed specification of all components developed during the four iterations released during the course of the project.

## 1.3. System Overview

### 1.3.1. System Requirements

The system requirements are set out in the D2.4 – Functional Requirements.

### 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) and D3.2 (Codebase on GitHub).

#### 1.3.2.2. Iteration 2 (WP3)

The project deliverables which relate directly to iteration 2 are D3.3 (Prototype) and D3.4 (Technical Integration Progress Report). There is also an internal deliverable, S3.5 (Codebase on GitHub).

#### 1.3.2.3. Iteration 3 (WP5)

The internal deliverables which relate directly to iteration 3 are S5.0 (Prototype) in month 24 and S5.0.1 (Codebase on GitHub) in Month 25.

#### 1.3.2.4. Iteration 4 (WP5)

Iteration 4, the final version of the ECK, is related to a number of deliverables. These are:

- D5.1 (Production version)
- D5.2 (Integration Status Report)
- D5.3 (Technical Documentation)
- D5.4 (Forward Plan)

There is also an internal deliverable S5.5 (Final Codebase on GitHub).

# 2. Design Considerations

## 2.1. Assumptions and Dependencies

Work Package 4 is dependent on the outputs of Work Packages 3 and 5 to produce its deliverables. Therefore, the following functionality was required for testing in the specified iteration:

| | |
|---|---|
| • Data Selection and Transformation | Iteration 1 |
| • Management Overview of status and data publication | Iteration 2 |
| • 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) were fixed in the description of work and can only be changed with the approval of the project monitoring officer.
- Each functional area should 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
- The specification for each functional component is designed to 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 followed 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. 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.

Development tasks in each iteration were assigned numbers relating to the area of functionality being developed. As the iterative development progressed, some functional areas were merged into a single module and other functions split across modules. It is therefore important to note that there is not a simple 1:1 mapping between functionality and modules.

# 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.

The high level architecture was designed to meet the following principles:

- The overall architectural style complies with established principles for service oriented architectures[1]
- The ECK comprises a set of modular components
- The components may be implemented locally or externally
- Functionality which is closely related to exist functionality within a CMS should be embedded within the CMS
- Components should expose (machine) interfaces to other components in a consistent fashion
- User interfaces, where necessary, will be embedded within the CMS and will be consistent with the look and feel of the individual CMS
- As well as the CMS, ECK components will interact with, and may be embedded in, aggregators
- Some ECK components will interact with aggregators or directly with Europeana

Figure 1is 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 a number of modules. These modules are summarised in Table 1.

As can be seen from Figure 1, some ECK modules have been incorporated into the aggregators used within the project. ECK functionality has also been implemented in other components of the ecosystem, such as middleware, but these variants have been left out of the diagram for simplification purposes.

[1] http://en.wikipedia.org/wiki/Service-oriented_architecture

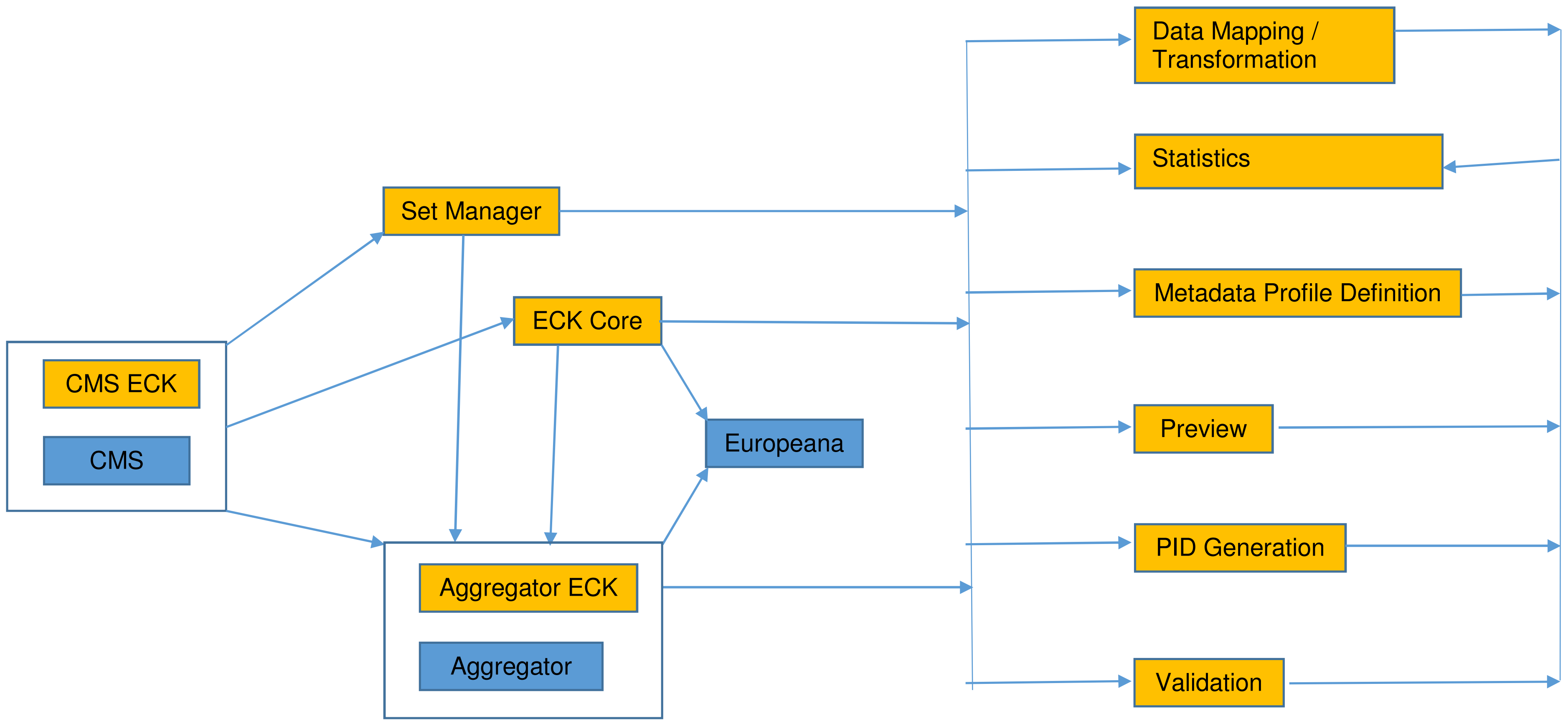


**Figure 1: Representation of the overall ECK architecture and its communication with the CMS and aggregator / Europeana**

Notes:

1. CMS / CMS ECK must contain either an OAI-PMH Repository or Data Push Client (Sword v1), management information and content re-ingestion functionality
2. Aggregator ECK contains an OAI-PMH client and repository, a Data Push Server (Sword v1), management information and content re-ingestion functionality
3. All modules document and implement their own persistence

## 3.2. ECK Modules

The table below shows the ECK modules and describes the functionality they deliver. These functional areas were identified by analysis of the requirements specification in conjunction with the high level design principles set out in section 3.1.

| Module | Functional Description |
|---|---|
| CMS: ECK supporting functionality | All functionality that must be provided within or alongside a CMS in order to meet the functional requirements set out in D2.4, including Data Push and/or OAI-PMH repository and content re-ingestion. |
| ECK Core | The gateway to the various modules, so supports all interfaces that all the other modules support. |
| Metadata Definition | Provides definitions for metadata elements within profiles and error messages in the different languages |
| Data Mapping / Transformation | Converts between formats, e.g. Native format to LIDO or LIDO to EDM |
| Preview | Provides a preview of how the record will be rendered in the Europeana user interface |
| PID Generation | Provides a persistent id for the record |
| Set Manager | Manages the sets and records |
| Statistics | Provides statistics about the performance of modules |
| Validation | Validates the record to ensure it conforms to a provided profile (of either LIDO or EDM). |
| Aggregator: ECK supporting functionality | Accepts data via data push or OAI-PMH, supplies data to Europeana via OAI-PMH and provides management information and enriched records for re-ingestion |

**Table 1: Summary of ECK functionality**

## 3.3. General Implementation and Integration

After consideration of the complexities of code-based integration, it was decided that direct access to the code library (or libraries) would make the ECK much more difficult to maintain and also to integrate with systems due to the requirement of different language-specific core and modules. It was therefore decided to deliver the ECK based on HTTP access and interaction between functional components, where necessary using language specific SDKs to wrap the HTTP calls. This is illustrated below.

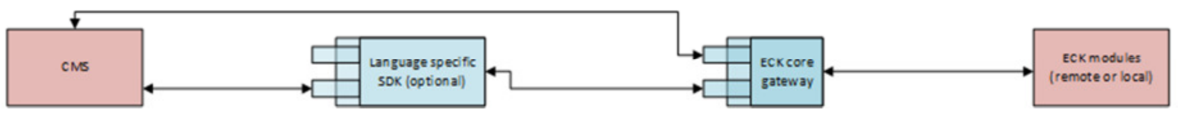

CMS
Language specific SDK (optional)
ECK core gateway
ECK modules (remote or local)

**Figure 2: Overall implementation / deployment overview enabling SDK access or direct HTTP access from the CMS code to the ECK code as required by each individual implementation scenario**

### 3.3.1. Consistent URL

As all the components expose a RESTful interface, all references to records within the system should conform to a consistent URL pattern. The pattern is:
*/<module>/<provider>/<Set, Group, Batch>/<action>/<record id>?parameters*

Where:

| | |
|---|---|
| **<module>** | is the root path for the module being called (eg. Set, PIDGeneration, Validation, Preview/Template) |
| **<provider>** | is the provider code for the owner of the record(s) |
| **<Set, Group, Batch>** | is the set / group or batch the record(s) belongs to |
| **<action>** | is the action to be performed, which will obviously vary between modules |
| **<record id>** | is the id of the record that an action needs to be performed against, this will not be required for all actions, for some modules this may not be required at all. |

If a single record is supplied as a parameter then it is assumed to be a single operation and the result is expected to be returned immediately.

If a zip file is supplied then it is assumed to be a batch operation, the module will return immediately with the http code 202 (Accepted).

Once a batch has been accepted then the action “status” can be called on that batch to retrieve the status of the operation (complete, in progress, queued etc.).

When the status action reports that the batch operation has been completed a “fetch” action can be performed against the batch to retrieve the results in a zip file.

It could be that even though only 1 record is supplied the module might want to treat it as a batch operation in which case the client will need to accept this.

If batch is not supported by the module, then it should return the http error code 501 (Not Implemented).

To obtain statistics form a module then the action “statistics” can be used, if the module has used the statistics module to store its statistics then the information can be retrieved from there, otherwise it will need to manage its statistics in some manner, if statistics are not supported by the module then it should return the http error code 501.

## 3.4. Illustrative workflow

The functional components of the ECK have been designed to allow a flexible range of deployment options and many different scenarios for invocation and orchestration between components. Table 3 below illustrates a simple workflow for a typical simple ECK implementation. It identifies the three main systems involved:

| | |
|---|---|
| **CMS** | The source Collections Management System or other supplying system |
| **AGG** | The aggregator (Culture Grid or project Dark Aggregator) |
| **EUR** | Europeana |

Note that all ECK functions may be called directly or via the ECK Core. CMS vendors may choose either route. All ECK functions accessed by Aggregators within the project are accessed via the ECK core. For simplicity of presentation, ECK core is not included in the table below.

| Step | Systems involved | Action |
|---|---|---|
| 1 | CMS | CMS user invokes **CMS:ECK** module to produce a LIDO representation of a batch of metadata records using **PID Generation** (if required) and one of **Data Mapping** or **Data Transformation**. Optionally, messages returned from modules are interpreted using **Metadata Definition**. |
| 2 | CMS | CMS user invokes **CMS:ECK** to call **Validation** and **Preview** for a sample of LIDO records. |
| 3 | CMS – AGG | CMS user invokes **CMS:ECK** to make records available to Aggregator using either native functionality or calling **Set Manager** to provide **OAI-PMH** or **Data Push** functions. |
| 4 | AGG | Aggregator receives LIDO records from CMS and calls **Validation** then **Data Transformation** to produce a batch of EDM records associated with a data provider. |
| 5 | AGG | Optionally, a data provider staff member accesses Aggregator user interface to view **Management Information, Statistics** and call **Preview** for a sample of records |
| 6 | AGG – EUR | Aggregator uses native OAI-PMH functionality to make records available to Europeana |
| 7 | EUR | Europeana ingests records and makes **Management Information** available |
| 8 | AGG - EUR | Aggregator requests **Management Information** from Europeana via **ECK Core** |
| 9. | AGG - EUR | Aggregator retrieves updated records from Europeana and generates an Enrichment record |
| 10 | CMS - AGG | CMS requests **Management Information** from Aggregator and Europeana via **ECK Core** |
| 11 | CMS – AGG | CMS requests enrichment records from Aggregator via **ECK Core** |

**Table 2: Illustrative ECK workflow (iteration 2)**

## 3.5. Other Considerations

### 3.5.1. Introduction

This section details other considerations that have been taken into account when designing the technical architecture and requirements.

### 3.5.2. Functional, High Level and Non-functional Requirements

Coverage of requirements by each module is outlined in Annexes 2, 3 and 4.

### 3.5.3. Extensibility

The architecture of the ECK is designed to be extensible to accommodate both changes to the requirements of Europeana and also to allow the code base to be used to supply data to other services.

### 3.5.4. Other Tools

There are already many existing tools that implement parts of the functionality required by the ECK. When specifying each module, the responsible developers first reviewed existing tools and utilized them where appropriate.to the extent they can be used or further developed as part of the ECK.

### 3.5.5. Other Required Modules

The ECK modules outlined in this document are designed to satisfy all the requirements set out in D2.4. However, it is acknowledged that, in future, there may be a need to add new modules or change existing ones to meet new and changing requirements.

# 4. Detailed Module Design

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

There is a section for each functional area originally identified from analysis of the functional requirements. Where functionality has been merged or split across modules this is noted in the relevant section.

## 4.1. CMS: ECK supporting functionality

### 4.1.1. Summary

This section represents functionality that must be provided within the main CMS in order for the CMS to be classed as ECK compliant. It includes functions such as selecting and exporting records in the relevant metadata profile to an aggregator, reporting on the state of record selection/ export and managing the re-ingestion of enriched records.

### 4.1.2. 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.

### 4.1.3. Technical Design

#### 4.1.3.1. 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.

#### 4.1.3.2. 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.

#### 4.1.3.3. Implementation Expectations

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

## 4.2. ECK Core

### 4.2.1. Summary

This represents the gateway to the ECK functionality as it is integrated into the various CMS systems. It is envisaged that this will be the interface that the CMS communicates directly with and will accept data from the 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.

### 4.2.2. 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 is exposed to the CMS using a HTTP RESTful interface.

The ECK core is stateless as it is essentially a gateway to the other components of the ECK.

### 4.2.3. Technical Design

#### 4.2.3.1. Interfaces

This provides an interface to the methods within the ECK, so that the CMS will not need to know where the different modules are installed, as these will be configured in the core.

The core will support all the API calls for all the modules that are expected to be called by the CMS, so using the set manager as an example, an API call to the set manager takes the form:
*/Set/<provider>/<set name>/<action>/<record id>?parameters*

for redirection though the gateway this would take the same form so instead of being:
*http://server/ECKSetManager/Set/<provider>/<set name>/<action>/<record id>?parameters*

it becomes:
*http://server/ECKCore/SetManager/Set/<provider>/<set name>/<action>/<record id>?parameters*

The list of path mappings for each module are as follows:

| Module | Path |
|---|---|
| Data Mapping | DataMapping |
| Data Transformation | DataTransformation |
| Statistics | Statistics |
| Metadata Profile Definition | Definition |
| PID Generation | PIDGeneration |
| Preview | Preview |
| Set Manager | SetManager |
| Validation (monguz) | Validation |
| Validation (semantika) | Validation2 |

For any modules not listed in the above table, it is not perceived that there will be a need for them to be called by the CMS.

In addition to the above modules there is also an aggregator interface built into ECK Core. This interface is capable of talking to any aggregator (e.g. Europeana) which supports the requested action.  If an aggregator does not support the action, or ECK Core does not support the action for that aggregator, then http status 400 (Bad Request) will be returned with an appropriate explanatory message.

The URL format for the aggregator interface is:
*/Aggregator/<aggregator>/<action>/<parameter1>/<parameter2>?parameters*

Where:
**aggregator** is the aggregator you want to communicate with (eg. CultureGridLive, CultureGridTest, DarkAggregator, Europeana, SetManager)

**action** is the action to be performed against the aggregator, the possible actions are:

| | |
|---|---|
| enrichmentRecord | returns the enriched record for the given record id and set |
| search | returns all the records for the supplied provider and collection |
| statistics | returns the statistics for the supplied provider and collection |

**parameter1** and **parameter2** vary depending on the action and are as follows:

**enrichmentRecord** action:

| | |
|---|---|
| parameter1 | set identifier |
| parameter2 | record identifier |

**search** and **statistics** actions:

| | |
|---|---|
| parameter1 | provider identifier |
| parameter2 | collection identifier |

The additional parameters that are supported by the actions are:

| Action | Parameter | Details |
|---|---|---|
| enrichmentRecord | **lidoRecID** (not supported by europeana) | can be used when the aggregators record identifier is unknown (assuming the lido record was uploaded to that aggregator) |
| search | **start** | where to start returning records from in the result set (1 is the first record) |
| | **rows** | number of records to return per request.  Note that an aggregator may enforce its own maximum.  Therefore it is not safe to assume the end of the result set has been reached just because less records are returned than requested. If an aggregator returns no records then it is safe to assume the end of the result set has been reached. |
| Statistics | none | |

### 4.2.3.2. 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 via the ECK core for use by the CMS which submitted the request.

#### 4.2.3.3. Implementation Expectations

It is expected that there will be one implementation of the core accessible via http. The http calls will also be able to be (optionally) wrapped in language specific SDKs to enable easier integration with external systems and to optionally insulate them from dealing with http calls directly. Such language specific SDKs can form an extensible integration library which can be contributed to as the ECK develops and becomes more widely used.

### 4.2.4. Online Documentation

The online documentation for the Core module can be found at http://euinside.k-int.com/ECKCore2/help/core

## 4.3. Metadata profile definition module

### 4.3.1. Summary

This module will allow for the provision of multilingual alternative labels for use by the source CMS when displaying information to end users. It has been designed to be used in two circumstances:

1. Defining labels for metadata elements. For example in profiles of LIDO, MARC or EDM, 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.

2. Enabling 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.

### 4.3.2. Assumptions

None

### 4.3.3. Technical Design

#### 4.3.3.1. Interfaces

The interface to this module is listed in the following table, all calls being an HTTP GET and the data will be returned in JSON:

| URL (pattern) | Function |
|---|---|
| /languages | Lists all available languages |
| /profiles | Lists all profiles in the default language |
| /profiles/<language> | Lists all profiles in the specified language |
| /profiles/<language>/<profile_identifier> | Lists all the definitions in the language specified for the given profile |
| /profiles/<language>/<profile_identifier>/<field> | Returns the definition for the specified profile and field combination for the language specified |
| /errors | Lists all errors in the default language |
| /errors/<language> | Lists all the error definitions in the specified language |
| /errors/<language>/<error_code> | Returns the specified error definition for the given language |

Where:

| | |
|---|---|
| <language> | is the ISO code for the language you are interested in |
| <profile_identifier> | is the identifier for the profile you are interested in |
| <field> | is the field you want the definition for |
| <error_code> | is the error you want the definition for |

##### 4.3.3.2. Conformance Criteria

This module will be deemed to conform to the specification when it exposes each of the above defined interfaces for use by the CMS components.

##### 4.3.3.3. Implementation Expectations

It is expected that this module will be shared throughout the different implementations and therefore there should be only one installation that holds all the different interpretations of the messages.

### 4.3.4. Online Documentation

The online documentation for the Metadata Definition module can be found at http://euinside.k-int.com/ECKCore2/Definition

## 4.4. Persistence module

From iteration 2 onwards there will no longer be a persistent module. Each module will provide, describe and manage its own persistent layer. The persistence functionality that was part of Iteration 1 is now part of the Set Manager.

## 4.5. PID Generation module

### 4.5.1. Summary

This module is in charge of creating PIDs for those records which do not already have suitable identifiers. 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.

### 4.5.2. Assumptions

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

It is expected that this module will be shared throughout the different implementations and therefore there should be only one installation that holds all the different interpretations of the messages.

### 4.5.3. Technical Design

#### 4.5.3.1. Interfaces

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

- Generate PID
- Reverse lookup PID (return constituent parts)

#### 4.5.3.2. Conformance Criteria

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

#### 4.5.3.3. Implementation Expectations

It is expected that this module will be shared throughout the different implementations and therefore there should be only one installation.

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:

- ISIL - http://biblstandard.dk/isil/
- MuseumID - http://museumid.net
- CIDOC ID recommendations - http://bit.ly/CIDOC-IdRecommendations

### 4.5.4. Online Documentation

The online documentation for the PID Generation module can be found at http://euinside.k-int.com/ECKCore/static/docs/Semantika_EU_Inside_PID_Generation_WS.pdf

## 4.6. Preview module
### 4.6.1. 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 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.

### 4.6.2. 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. Some implementations may prefer to expose the preview directly on a website to be linked to, while others may prefer full control and as such require the preview bundle to be returned. 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.

### 4.6.3. Technical Design
#### 4.6.3.1. 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

#### 4.6.3.2. 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

#### 4.6.3.3. Implementation Expectations

It is expected that this module will be shared throughout the different implementations and therefore there should be only one installation.

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.

### 4.6.4. Online Documentation

The online documentation for the Preview module can be found at http://euinside.k-int.com/ECKCore/static/docs/preview-rest-api-MON.pdf

## 4.7. Validation module

### 4.7.1. Summary

This module provides validation functionality for the ECK. It receives one or more metadata documents. 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[2]
- 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 4.3.

### 4.7.2. Assumptions

None

### 4.7.3. Technical Design

#### 4.7.3.1. Interfaces

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

- One by one validation
- Batch validation

#### 4.7.3.2. Conformance Criteria

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

#### 4.7.3.3. Implementation Expectations

It is expected that there will probably be one validation module implementation.

### 4.7.4. Online Documentation

The online documentation for the validation modules can be found at http://euinside.k-int.com/ECKCore/static/docs/validation-rest-api-common.pdf

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

## 4.8. Set Manager

### 4.8.1. Summary

The set manager allows a controlled way for sets to be managed, harvested, pushed and for actions to be performed in a consistent manner, actions can be performed on the set as a whole or on individual records.

There will be 2 versions of a set, a live set which is the version that will be posted to Europeana and the working set where changes are made, the working set becomes the live set when the action “commit” is performed on the set.

To conform with the OAI-PMH protocol, records are not deleted, they are only ever marked as deleted.

### 4.8.2. Assumptions

None.

### 4.8.3. Technical Design

#### 4.8.3.1. Interfaces

The URL for the set module will be:
*/Set/<provider>/<set name>/<action>/<record id>?parameters*

Where:

| | |
|---|---|
| **<provider>** | Is the code for the provider of the data (it will be restricted by client IP address as to which machines can provide/request data for a provider) |
| **<set name>** | is the set that is to be manipulated or information is to be provided for, if the set does not exist, it will be created, for compatibility with iteration 1 the set name of “default” will be created for any data persisted in iteration 1. |
| **<action>** | is the action to be performed on the set |
| **<record id>** | is the record to be actioned, this is not applicable for all actions and will be ignored if supplied where it is not relevant. |

The possible parameters are:

| Parameter | Meaning | Value |
|---|---|---|
| collection | The collection identifier required for data push | |
| delete | A comma separated list of records to be deleted from the set | |
| deleteAll | Mark all records in the set as being deleted | Yes / No |
| europeanaId | The europeana id for this provider | |
| historyItems | The number of history that should be returned as part of the status message | default 20 |
| live | If set to yes the live set will be returned, otherwise the working set be returned | |
| password | The password required for data push | |
| provider | The provider identifier required for data push | |
| records | A posted zip file that contains the records that are to be added to the set or update existing records | Valid records |
| setDescription | A description that can be used for the set | |
| status | The status of the records to be returned by the list action | all (default)<br>pending<br>error<br>valid<br>deleted |
| swordURL | The url to be used for data push | |
| username | The username required for data push | |

A set could have one of the following statuses:

| Status | Meaning | Applies to Set |
|---|---|---|
| Dirty | The set has been updated since the last commit | Working |
| Validating | Validating prior to a commit | Working |
| Commit | Commit is in progress | Live, Working |
| Committed | No changes since the last commit | Live, Working |
| Error | For all scenarios where an error has occurred and a retry option is no longer viable | Live, Working |

The following actions can be performed on a set:

| Action | Details | Data Returned | Record Id Valid | Valid Parameters |
|---|---|---|---|---|
| commit | Informs the system to perform validation on all records in the set that have yet to be validated, makes all valid record in the set available to Europeana for harvesting.<br>If records are also supplied it performs an update action first.<br>The action is queued and therefore returns immediately, use the *status* action to find more details. | None | No | delete<br>deleteAll<br>records |
| edit | Edits the details associated with the set | None | No | collection<br>europeanaId<br>password<br>provider<br>setDescription<br>swordURL<br>username |
| list | Returns the list of identifiers from the working set and whether the record is valid or not. | JSON array of objects that contain:<br>• cms id<br>• persistent id<br>• validation status (OK, Error or Pending) | No | live<br>status |
| preview | Calls the preview module for the specified record id | same as returned by preview module | Yes | |
| push | Pushes the committed data to the specified sword server, the server is specified using the swordURL parameter. The destination username, password and onBehalf is specified on the provider record and the collectionId for the destination is specified on the ProviderSet | None | No | |
| record | Returns the record for the specified record id from the working set as xml. | the original data supplied for the record | Yes | |

| Action | Details | Data Returned | Record Id Valid | Valid Parameters |
|---|---|---|---|---|
| | If no record id is supplied then all records in the working set are returned as a zip file. | | | |
| statistics | Provides statistics for the set | JSON object that returns the following:<br>• The provider code<br>• The collection code<br>• The set description<br>• Number of records accepted (live)<br>• Number of pending records<br>• Number of rejected records<br>• Total number of records<br>• Most recent europeana statistics | No | |
| status | Provides the status of the set | JSON object that contains the following:<br>• When the set was created<br>• When the harvest / post became available<br>• The status of the live and working sets<br>• How many records in the working set<br>• How many records in the live set<br>• How many records are valid<br>• How many records are waiting to be validated<br>• How many records are invalid<br>• Array of pending actions | No | historyItems |
| update | Takes the supplied zip of records and updates the working set with these records.<br>The action is queued and therefore returns immediately, use the *status* action to find more details. | None | No | delete<br>deleteAll<br>records |
| validate | Return the validation errors for all invalid records, if the record_id is specified it will return just the errors for that | JSON array of objects that contain the | Yes | |

| Action | Details | Data Returned | Record Id Valid | Valid Parameters |
|---|---|---|---|---|
| | record | following:<br>• cms id<br>• array of objects containing<br>    • error code<br>    • additional information | | |

#### 4.8.3.2. Conformance Criteria

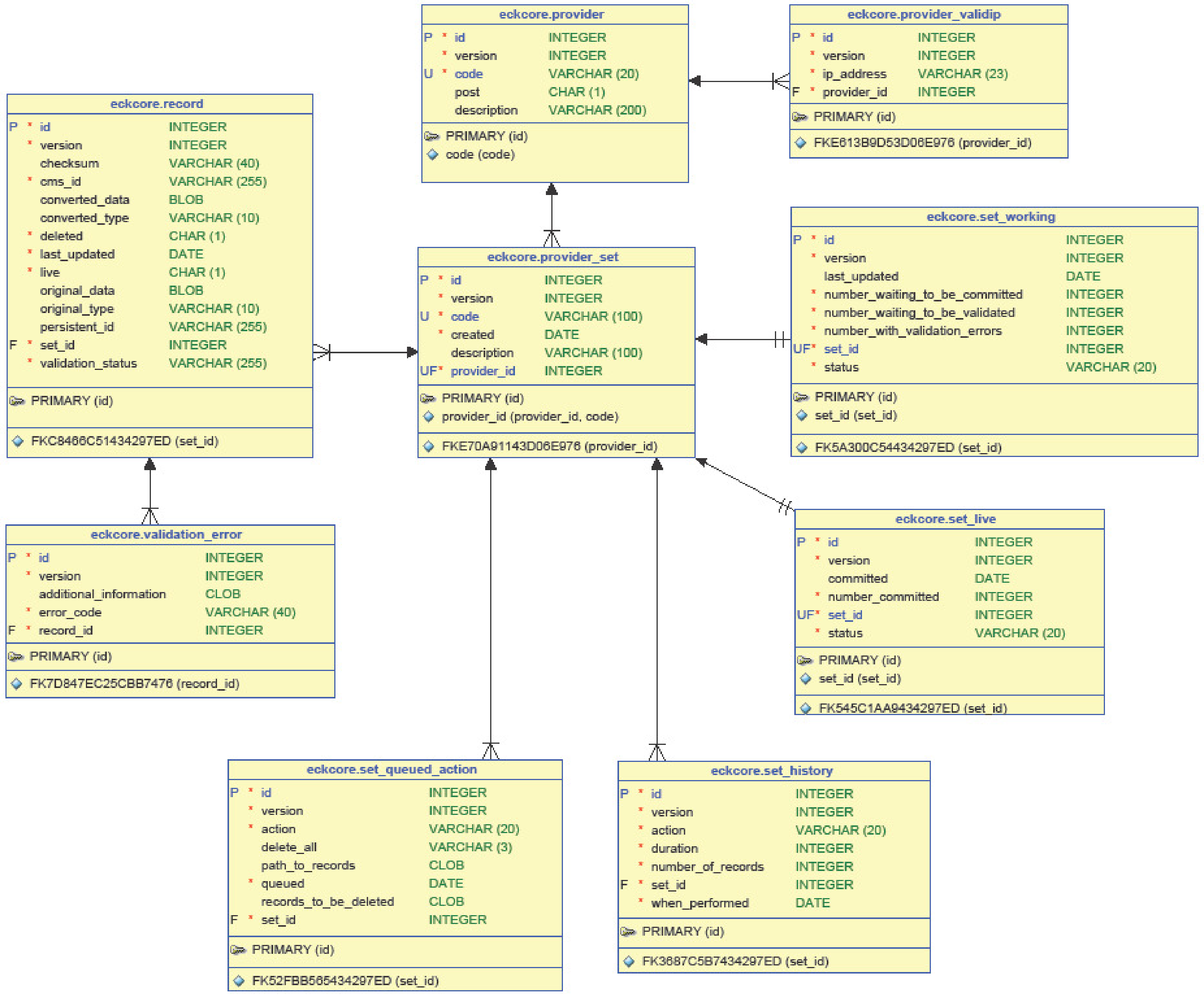
Implementation of this module is deemed to be complete when all the actions defined above are fully implemented.

#### 4.8.3.3. Implementation Expectation

There will be both local and remote implementations of this module.

#### 4.8.3.4. Persistence Implementation

The diagram below shows the persistence implementation for this module:


eckcore.provider
P * id INTEGER
* version INTEGER
U * code VARCHAR (20)
post CHAR (1)
description VARCHAR (200)
PRIMARY (id)
code (code)
eckcore.provider_validip
P * id INTEGER
* version INTEGER
* ip_address VARCHAR (23)
F * provider_id INTEGER
PRIMARY (id)
FKE613B9D53D06E976 (provider_id)
eckcore.record
P * id INTEGER
* version INTEGER
checksum VARCHAR (40)
* cms_id VARCHAR (255)
converted_data BLOB
converted_type VARCHAR (10)
* deleted CHAR (1)
* last_updated DATE
* live CHAR (1)
original_data BLOB
original_type VARCHAR (10)
persistent_id VARCHAR (255)
F * set_id INTEGER
* validation_status VARCHAR (255)
PRIMARY (id)
FKC8466C51434297ED (set_id)
eckcore.provider_set
P * id INTEGER
* version INTEGER
U * code VARCHAR (100)
* created DATE
description VARCHAR (100)
UF* provider_id INTEGER
PRIMARY (id)
provider_id (provider_id, code)
FKE70A91143D06E976 (provider_id)
eckcore.set_working
P * id INTEGER
* version INTEGER
last_updated DATE
* number_waiting_to_be_committed INTEGER
* number_waiting_to_be_validated INTEGER
* number_with_validation_errors INTEGER
UF* set_id INTEGER
* status VARCHAR (20)
PRIMARY (id)
set_id (set_id)
FK5A300C54434297ED (set_id)
eckcore.validation_error
P * id INTEGER
* version INTEGER
additional_information CLOB
* error_code VARCHAR (40)
F * record_id INTEGER
PRIMARY (id)
FK7D847EC25CBB7476 (record_id)
eckcore.set_live
P * id INTEGER
* version INTEGER
committed DATE
* number_committed INTEGER
UF* set_id INTEGER
* status VARCHAR (20)
PRIMARY (id)
set_id (set_id)
FK545C1AA9434297ED (set_id)
eckcore.set_queued_action
P * id INTEGER
* version INTEGER
* action VARCHAR (20)
delete_all VARCHAR (3)
path_to_records CLOB
* queued DATE
records_to_be_deleted CLOB
F * set_id INTEGER
PRIMARY (id)
FK52FBB565434297ED (set_id)
eckcore.set_history
P * id INTEGER
* version INTEGER
* action VARCHAR (20)
* duration INTEGER
* number_of_records INTEGER
F * set_id INTEGER
* when_performed DATE
PRIMARY (id)
FK3687C5B7434297ED (set_id)

#### 4.8.4. Online Documentation

The online documentation for the Set Manager module can be found at http://euinside.k-int.com/ECKCore2/SetManager

## 4.9. Statistics

### 4.9.1. Summary

The statistics module allows users to track usage off the modules. This data, along with data from the set manager, can be used to provide feedback to data providers. During development and testing, it can also be used to determine where and when bottlenecks occur.

### 4.9.2. Assumptions

None.

### 4.9.3. Technical Design

#### 4.9.3.1. Interfaces

The following interfaces will be available for the statistics module:

1. */Statistic/update/<module>?Parameters*

The parameters that are to be supplied are:

| **Parameter** | **Meaning** |
|---|---|
| itemsProcessed | The number of items that were processed |
| dateTime | Date and time of when the processing started, format: ISO8601, RFC3339 |
| numberSuccessful | The number of items that were successfully processed |
| numberFailed | The number of items that failed to be processed |
| Duration | The length of time in milliseconds that it took to process these records |

There is no data returned for this call.

2. */Statistic /query/<module>/<type of query>?limit=<maximum number of items to return>&duration<the base line for the duration>*

The parameters that can be supplied are

| **Parameter** | **Meaning** |
|---|---|
| Limit | Maximum number of items to return |
| Duration | The minimum duration of time that we are interested in |

The data returned is a JSON array of objects that contain the following:

- The date and time of when the processing started
- The number of items processed
- The number of successfully processed items
- The number of items that failed processing
- The duration in milliseconds of the time it took to perform the processing

The query types that will be acceptable are:

| Type | Meaning |
|---|---|
| Slowest | Return the longest running calls (based on the average time taken to process 1 item) |
| Status | Returns the following<br>• Total number of items processed<br>• Total time taken to process the items<br>• Fastest time to process an item<br>• Slowest time to process an item<br>• Average time taken to process the items |

#### 4.9.3.2. Conformance Criteria

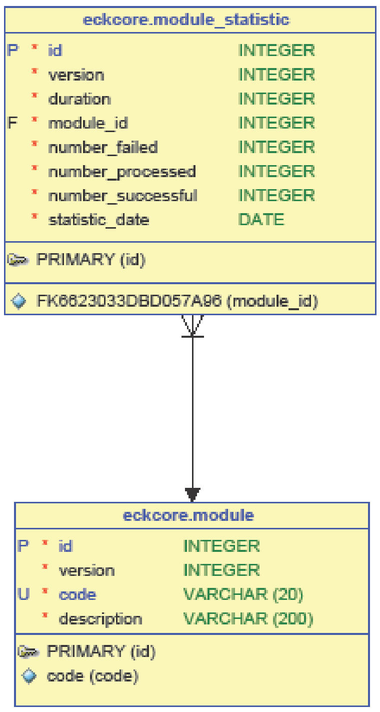
Implementation of this module is deemed to be complete, when the interfaces defined above are fully implemented

#### 4.9.3.3. Implementation Expectation

This should be installed with every implementation, so statistics can be gathered

#### 4.9.3.4. Persistence Implementation

The diagram below shows the persistence implementation for this module.



#### 4.9.4. Online Documentation

The online documentation for the Statistics module can be found at http://euinside.k-int.com/ECKCore2/Statistics/

## 4.10. Data Mapping / Transformation

### 4.10.a. Data Mapping

#### 4.10.a.1 Summary

The Data Mapping module will map a record from one format to another using the supplied rules.

#### 4.10.a.2 Assumptions

None.

#### 4.10.a.3. Technical Design

##### 4.10..a.3.1 Interfaces

The url for the module will be */DataMapping/<provider>/<batch>/<action>?parameters*

Where:

| | |
|---|---|
| **<provider>** | **i**s the code for the provider of the data |
| **<batch>** | is the name for this batch. |
| **<action>** | is the action to be performed on this batch |

The possible parameters are:

| **Parameter** | **Meaning** |
|---|---|
| Record | An xml file that requires mapping |
| Records | A zip file that contains the records that require mapping |
| sourceFormat | The format of the source record |
| targetFormat | The format of the target of the mapping |
| mappingRulesFile | A csv file that contains the rules required to perform the mapping |
| requestId | The request identifier returned by the Transform action, that is to be used for the Status and Fetch actions |

The possible actions are:

| **Action** | **Meaning** |
|---|---|
| Fetch | Fetches the results for the batch submission |
| List | Returns a list of the supported formats |
| Status | Returns the status of the batch submission |
| Transform | Transforms the supplied record(s) |

The mapping will return records in XML, for the Fetch action the records will be returned in XML or in a zip file if they are submitted in a zip file.

If a format is not supported then an appropriate http error code will be returned saying that it is unsupported.

#### 4.10.a.3.2. Conformance Criteria

Implementation of this module is deemed to be complete, when the interfaces defined above are fully implemented

#### 4.10.a.3.3. Implementation Expectation

There should only be a need for 1 implementation of this module.

#### 4.10.a.3.5. Online Documentation

The online documentation for the Data Mapping module can be found at http://euinside.k-int.com/ECKCore/static/docs/ManualDataMappingTransformationService.doc

## 4.10.b. Data Transformation

### 4.10.b.1 Summary

The Data Transformation module will transform a record from one format to another. In iteration 2, development will concentrate on LIDO to EDM mapping. In subsequent iterations, the module may be extended to provide mappings from a range of input formats to LIDO for use in implementations where this conversion is not handled by the ECK CMS supporting functionality module.

### 4.10.b.2 Assumptions

For iteration 2, all implementations will be able to provide LIDO to this module. LDO to EDM mapping will be based on mappings provided by other Europeana family projects,

### 4.10.b.3. Technical Design

#### 4.10..b.3.1 Interfaces

The url for the module will be */DataMapping/<provider>/<batch>/<action>?parameters*

Where:

| | |
|---|---|
| **<provider>** | is the code for the provider of the data |
| **<batch>** | is the name for this batch. |
| **<action>** | is the action to be performed on this batch |

The possible parameters are:

| Parameter | Meaning | Value |
|---|---|---|
| record | An xml file that requires mapping | A record that is valid for the source format |
| records | A zip file that contains the records that require transforming | Zipped records that are valid for the source format |
| requestId | The request identifier returned by the Transform action, that is to be used for the Status and Fetch actions | |
| sourceFormat | The format the supplied record is in | Any valid format that the module supports conversions from, in iteration 2 this will only be LIDO, if this is omitted then LIDO will be defaulted |
| targetFormat | The format that that the record will be returned in | Any valid format that the module supports conversions to, in iteration 2 this will only be EDM, if this is omitted then EDM will be defaulted |

The possible actions are:

| Action | Meaning |
|---|---|
| Fetch | Fetches the results for the batch submission |
| List | Returns a list of the supported formats |
| Status | Returns the status of the batch submission |
| Transform | Transforms the supplied record(s) |

The mapping will return records in XML, for the Fetch action the records will be returned in XML or in a zip file if they are submitted in a zip file.

If a format is not supported then an appropriate http error code will be returned saying that it is unsupported.

#### 4.10.b.3.2. Conformance Criteria

Implementation of this module is deemed to be complete, when the interfaces defined above are fully implemented

#### 4.10.b.3.3. Implementation Expectation

There should only be a need for 1 implementation if this module.

#### 4.10.b.3.4. Online Documentation

The online documentation for the Data Mapping module can be found at http://euinside.k-int.com/ECKCore/static/docs/ManualDataMappingTransformationService.doc.

## 4.11. OAI-PMH Repository

### 4.11.1. Summary

This is no longer a separate module. The Set Manager module now provides the OAI-PMH repository. Some implementations will incorporate a local OAI-PMH repository into the ECK CMS supporting functionality.

## 4.12. Data Push

### 4.12.1. Summary

This is no longer a separate module. The Set Manager module now provides a repository to support the data push service. Some implementations will incorporate a data push (SWORD v1) client into the ECK CMS supporting functionality.

## 4.13. Aggregator/ ECK Supporting Functionality (was Content Re-ingestion)

### 4.13.1. Summary

The Aggregator ECK performs the following functionality within the ECK:

- a data push server
- an OAI-PMH Client
- an OAI-PMH Repository
- generating the enrichment record for the content re-ingestion process
- providing management information
- managing Aggregator ECK Workflow.

#### 4.13.1.1. Data Push Server

This server complies too the Sword v1 protocol, so it can accept records from a CMS that has implemented a Sword v1 client.

#### 4.13.1.2. OAI-PMH Client

The client complies too the OAI-PMH v2 protocol, so it can harvest records from a CMS that has implemented an OAI-PMH Repository v2.

#### 4.13.1.3. OAI-PMH Repository

The repository complies too the OAI-PMH v2 protocols so it can make the records available to Europeana.

#### 4.13.1.4. Providing Management Information

Returns statistical management information including the status of record processing (number of records accepted, pending, rejected, published, etc.) from both the aggregator and Europeana using the ‘statistics’ call described in section 4.2.3.1.

##### 4.13.1.5. Generating the Enrichment Record for Content Re-ingestion

As part of content re-ingestion process the aggregator retrieves the published records from Europeana and generates an enrichment record that can be requested by the CMS.

##### 4.13.1.6. Managing Aggregator ECK Workflow

The aggregator provides a workflow for obtaining / receiving the records from the CMS, validating the records, transforming the records to EDM if they are valid, making the EDM records available to Europeana, generating the enrichment record and making it available to the CMS.

### 4.13.2. Assumptions

None

### 4.13.3. Technical Designs

The module interacts with the ECK Core (in order to integrate with the Validation, Preview and Data Transformation modules) and also make the enrichment record available to the CMS. ECK Core interfaces are described in section 4.2.

### 4.13.4. Online Documentation

See the appropriate section within this document for the online documentation about the modules used.

The documentation for aggregator section of the ECKCore can be found at http://euinside.k-int.com/ECKCore2/help/aggregator

The OAI-PMH v2 protocol can be found at http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm

The Sword v1 protocol can be found at http://swordapp.org/sword-v1/

# 5. Scheduling & Issues

## 5.1. Development Schedule

The development schedule for each area of functionality, along with associated task numbers is described in the table below:

| Functional Area | Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 |
|---|---|---|---|---|
| CMS: ECK supporting functionality | **3.1.1** | 3.2.1 | 5.1.1 | 5.2.1 |
| ECK Core | **3.1.2** | 3.2.2 | 5.1.2 | 5.2.2 |
| Metadata Profile Definition | **3.1.3** | 3.2.3 | 5.1.3 | 5.2.3 |
| Persistence | **3.1.4** | | | |
| PID Generation | **3.1.5** | 3.2.5 | 5.1.5 | 5.2.5 |
| Preview | **3.1.6** | 3.2.6 | 5.1.6 | 5.2.6 |
| Validation | **3.1.7** | 3.2.7 | 5.1.7 | 5.2.7 |
| Set Manager | | **3.2.8** | 5.1.8 | 5.2.8 |
| Statistics | | **3.2.9** | 5.1.9 | 5.2.9 |
| Data Mapping/ Transformation | | **3.2.10** | 5.1.10 | 5.2.10 |
| OAI-PMH Repository | | **3.2.11** | 5.1.11 | 5.2.11 |
| Data Push | | **3.2.12** | 5.1.12 | 5.2.12 |
| Content Re-ingestion/ ECK Aggregator | | | **5.1.13** | 5.2.13 |

## 5.2. Outstanding Issues

The following issues could not be addressed within the timescales set for the production of this deliverable.

- How is ECK conformance to be defined and measured?

To be addressed as part of the forward plan (D5.4)

- How are features such as data push to supported by the Europeana infrastructure after the end of the project?

To be addressed as part of the forward plan (D5.4)

- How are tasks such as maintenance of provider codes and provision of aggregator services for institutions without access to another aggregator to be addressed after the end of the project?

To be addressed as part of the forward plan (D5.4)

- Is the approach to validation taken in the majority of implementations (i.e. validation of LIDO records but no validation of EDM) sufficient to guarantee delivery of high quality data to Europeana?

To be addressed by the coordinating partner in discussion with Europeana and other stakeholders.

# 6. Acceptance & Sign Off

## 6.1. Acceptance

Software components must meet the specified conformance criteria to be accepted. Content partners were responsible for verifying that the conformance criteria have been met. Acceptance testing took place as part of work package 4 and is reported on as part of that work package.

## 6.2. Sign Off

All internal and external deliverables were 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<br><br>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<br><br>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<br><br>D3.4 Technical Integration Report | D3.3 Prototype<br><br>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<br><br>D4.4 Content Re-ingestion Report<br><br>D4.5 Summative Evaluation Report<br><br>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<br>D4.4 Content Re-Ingestion Report<br>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<br>D5.3 Technical Documentation | D5.2 Integration Status Report<br><br>D5.3 Technical Documentation | D4.3(v2) Export Evaluation Report<br><br>D4.5(v2) Summative Evaluation Report |
| 30 | D5.4 Forward Plan | D5.4 Forward Plan<br><br>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 |

# Annex 2 Coverage of Functional Requirements

Key:

| | |
|---|---|
| □ | Must |
| 🟧 | Should |
| ▨ | Could |
| 🟥 | Won’t |

| Req | Task | Title |
|---|---|---|
| 01.01 | CMS | Export management |
| 01.02 | CMS<br>Management Information | Revision history |
| | | |
| 01.04 | CMS<br>PID | PID management |
| 01.05 | Re-ingestion | Enriched data management |
| 02.01 | CMS | Select multiple records |
| 02.02 | CMS | Select single record |
| 02.03 | CMS | Select records based on values |
| 02.04 | CMS | Boolean operators |
| 02.05 | CMS | Indication of selected fields |
| 02.06 | | Selecting within record |
| 02.07 | CMS | Reuse saved queries |
| 02.08 | | Manage multiple selections |
| 02.09 | | Standardised selection filters |
| 03.01 | Transformation | Automatic EDM mapping |
| 03.02 | Preview | Preview mapping |
| 03.03 | Mapping | Editable mapping |
| 03.04 | CMS<br>Mapping | Mapping feedback |
| 03.05 | Mapping | Saving mapping |
| 03.06 | Mapping<br>Metadata Profile Definition | Field explanations |
| 03.07 | Mapping | Automatic value insertion |

| | | |
|---|---|---|
| | Transformation | |
| 03.08 | CMS | Check digital asset availability |
| 03.09 | CMS<br>Mapping<br>Transformation | Thumbnail selection |
| 03.10 | CMS<br>Mapping<br>Transformation | Multiple assets |
| 03.11 | CMS<br>Mapping<br>Transformation | Defining media type |
| 03.12 | CMS<br>Mapping | Metadata field on IPR digital object |
| 03.13 | CMS<br>Mapping | Metadata field on IPR metadata |
| 03.14 | CMS<br>Mapping | Metadata field on IPR preview |
| 03.15 | CMS<br>Mapping<br>Transformation | Mark mandatory fields |
| 03.16 | CMS<br>Transformation | Choose default mapping |
| 03.17 | CMS | Automatic data suggestion |
| 03.18 | CMS | Target format selection |
| 03.19 | CMS<br>Mapping | Semantic data enrichment |
| 03.20 | CMS<br>Mapping | Conditional mapping |
| 03.21 | CMS<br>Mapping | Nested or grouped mapping |
| 03.22 | CMS<br>Mapping<br>Transformation | Intermediate format mapping |
| 03.23 | CMS<br>Mapping | Support for conditional truncation |
| 03.24 | CMS<br>PID | Apply PID |
| 03.25 | CMS<br>Mapping | Conditional field conversion |
| 04.01 | Validation | Validation |
| 04.02 | Validation | Feedback on validation |
| 04.03 | CMS<br>Validation | Edit invalidated fields |
| 04.04 | Validation | Automatic license validation |
| 04.05 | OAI-PMH<br>Data Push | Test ingestion |
| 04.06 | Validation | Align validation |
| 05.01 | CMS<br>Core | Auto supply |

| | | |
|---|---|---|
| | OAI-PMH<br>Data Push | |
| 05.02 | CMS<br>Core<br>OAI-PMH<br>Data Push | Re-supply functionality for failed records |
| 05.03 | CMS<br>Core<br>OAI-PMH<br>Data Push | Schedule data supply |
| 05.04 | Mapping<br>Transformation<br>OAI-PMH<br>Data Push | Supply for 3rd party collaboration |
| 06.01 | Preview | Preview presentation Europeana |
| 06.02 | CMS<br>Core<br>OAI-PMH<br>Data Push | Withdraw records |
| 06.03 | CMS<br>Core<br>OAI-PMH<br>Data Push | Update published records |
| 06.04 | Management<br>Information | Publication indication |
| 06.05 | Management<br>Information | Automatic publication alert |
| 07.01 | CMS<br>Management<br>Information<br>Re-ingestion | Available enriched content alert |
| 07.02 | CMS | Accept/ decline enrichments (record level) |
| 07.03 | CMS<br>Re-ingestion | Automatic ingest of enriched data |
| 07.04 | CMS<br>Re-ingestion | Separate enriched content |
| 07.05 | CMS<br>Re-ingestion | Enriched IPR identification |
| 07.06 | CMS | Choose target ingest |
| 07.07 | CMS | Accept/ decline (field level) |
| 07.08 | Re-ingestion | PID enrichment |
| 07.09 | Re-ingestion | Pull option |
| 07.10 | Management<br>Information<br>Re-ingestion | Enriched data management |

## Annex 3 Coverage of High Level Requirements

| Req | Task | Title |
|---|---|---|
| HLR.01 | Implicit in architecture | Exchange of cultural data |
| HLR.02 | All | Contributing to Europeana |
| HLR.02 | CMS<br>Re-ingestion | Adding value to local collections |
| HLR.04 | CMS<br>Core | Data management |
| HLR.05 | CMS<br>Core<br>Management Information | Transparency |
| HLR.06 | CMS<br>OAI-PMH<br>Data Push | Choice of data pull or push to Europeana |
| HLR.07 | Mapping<br>Transformation<br>OAI-PMH<br>Data Push | Multiple targets (same as 05.04?) |
| HLR.08 | Implicit in architecture | Various routes |
| HLR.09 | CMS<br>Mapping | Contextualisation |
| HLR.10 | Implicit in architecture | Re-use available knowledge |
| HLR.11 | Implicit in architecture | Modular |
| HLR.12 | Implicit in architecture | Export-import |
| HLR.13 | Implicit in architecture | API |
| HLR.14 | To be decided | Communication of changes |
| HLR.15 | CMS<br>Management Information | Version tracking |
| HLR.16 | CMS | Changing and saving of settings |

## Annex 4 Coverage of non-functional Requirements

| Req | Task | Title |
|---|---|---|
| NFR.01 | T5.4 | Sustainable and persistent workflow |
| NFR.02 | T5.4 | Label for CMS software |
| NFR.03 | CMS | User friendly |
| NFR.04 | | Auto-update |
| NFR.05 | Implicit in architecture | Make cultural heritage available to digital services |
| NFR.06 | CMS<br>T5.3 | User manual and training materials |
| NFR.07 | CMS<br>Metadata profile Definition<br>T5.3 | Multilingual support and documentation |
| NFR.08 | Implicit in architecture | Flexibility and adaptability |
| NFR.09 | Implicit in architecture | Open standards |
| NFR.10 | Implicit in architecture | Re-use existing tools |
| NFR.11 | Implicit in architecture | Modular (same as HLR.11) |
| NFR.12 | Implicit in architecture<br>T5.4 | Easy adaptability |
| NFR.13 | CMS | Simplicity |
| NFR.14 | T5.4 | Public-private partnership |
| NFR.15 | Implicit in architecture | Master-slave |
| NFR.16 | T5.4 | Organisation embedding |